Internship report

Study of AI and Cloud technologies application for enhancement of communication

University of Strasbourg

2020

Vadym Hadetskyi

CSMI M2

Table of content

Introduction

This report aims to present the work conducted in the scope of the internship on the research of possibilities of AI-based methods to enhance communication in the domains of marketing strategy and user experience enhancement. The availability of machine learning and cloud technologies opens a lot of ways to improve and optimize numerous processes across various fields of enterprise activity, from production to customer relations. Thus, the goal of the given endeavor is to find out how the company may profit from development of those techniques.

The internship if offered by Vivialys Groupe — biggest supplier of housing in Grand Est region of France. Main workload is distributed among the projects that belong to research & development department of the enterprise, which are:

- Development of smart chatbot
- IBM Cloud infrastructure
- Definition of personality profiles
- Automatic configurator of apartment modules

Based on those, principal areas of work and technologies involved in the frame of the internship are:

- Unsupervised learning, clustering, dimensionality reduction;
- Text mining, NLP;
- Chatbot development;
- Application development, containers, docker;
- IBM Cloud;
- Node Red

First main objective of the project is to present a tool that is capable to generate an insight regarding the personality traits of a person, given their writing example as an input.

Second main objective is to develop a chatbot able to conduct a conversation with a potential client, to make database queries based on user's input and to return a corresponding result.

The Roadmap

The roadmap consists of a set of subtasks and subproblems of the project that have to be furnished to fulfill the principal objective, or that might yield better results. One must have in mind that some of them might change, be replaced or removed during the development of the project. Nonetheless, a roadmap serves as a guiding light at early stages of its evolution. For the moment, three principal branches of the development of project may be defined:

I. Identification of personality profiles

II. Chatbot development

III. Cloud Infrastructure

Subtask	Branch of the project	Estimated workload (hours)	Deadline	Completion
Chatbot prototype	Chatbot	25	End of June	1
Text Data mining	Personality	50	End of May	1
Exploration and learning of IBM Cloud	Cloud	32	End of May	1
Enterprise introduction and current projects	-	14	Mid June	1
Text Data cleansing	Personality	20	Start of June	1
Personality Insight	Personality	20	Start of June	1
Chatbot structure definition	Chatbot	16	Mid July	1

Subtask	Branch of the project	Estimated workload (hours)	Deadline	Completion
Entities, Intentions, Dialogs definition	Chatbot	24	End of July	1
Study of clustering methods	Personality	80	Start of August	1
Study of dimensionality reduction methods	Personality	42	End of July	1
Study of clustering evaluation techniques	Personality	32	Mid July	1
Data Visualization	Personality	24	End of August	1
Personality App Development	Personality	64	Start of September	1
App containerization	Cloud	24	Mid September	1
Chatbot deployment	Chatbot	24	Mid September	1
Chatbot-Database connection	Chatbot	48	Start of September	1
Cloud CLI and Kubernetes	Cloud	36	End of August	1

The workload estimations are, of course, approximate values of time needed to fulfill the objective. Some of those are expected time estimations, while others are approximate amounts of time spent for already finished tasks.

Completion column corresponds to the share of necessary work that is already done.

1. Context, objectives and evolution of the internship

Data is the fuel that drives both research and business to innovation. It creates new opportunities and even new domains for the first and opens previously unseen possibilities for the second. And very often the process of innovation is what creates a common ground for the two: business struggles to achieve competitive edge, while research always struggles for funding.

Thus, all the major companies across all the industries have been creating their "Research & Development" departments with a goal to gain an advantage over the competitors and to improve their own products and services. The given internship is being carried out in the R&D department of housing supplier company "Vivialys Groupe".

In the circumstances of the uncertainty and dynamic development of the modern world, it is nothing but reasonable to expect any project to undergo changes and to evolve. It is important to cherish the original idea, but also to keep the final purpose and to react to the changing environment based on the new available information.

Initially, the main objective of the internship was to develop with the use of artificial intelligence a smart chatbot, capable of detecting human's personality. The bot itself should offer a common functional of presenting an information, such as information about the company or its programs based on user's entry. Besides that, other key feature of the chatbot is the ability to perform queries to the available apartments database following user's criteria. All in all, first concept of the chatbot was meant to have following functional:

- Basic communication, information transmission;
- Apartments database querying;
- Personality identification.

However, with the passage of time it became clear that those functionalities are not compatible with each other. The very existence of chatbots revolves around standardization processes: to successfully mimic a conversation, all of the bot's

questions should be defined so that any possible response from a user is considered and expected, so it can be treated correctly.

On the other hand, personality identification loses meaning when applied to homogeneous, standardized instances. It is a challenging endeavor that demands significant amounts of data during training phase, as well as it has a strict requirements for the input in the production. It is said that, theoretically, machine learning algorithm may be trained to perform any task a human can, with sufficient data and well-thought approach to the training, of course. So, it is logical to ruminate on the amount of data that a human being needs to, say, classify other humans into extraverts and introverts. What if the goal is to also consider other human's needs, values and various traits of personality ?

The difficulty of the task and the data requirements scale up very quickly. Certainly this requirement exceeds few dozens of words which one can obtain with the help of chatbot. This is why, the initial objective was divided into two different, though connected projects: *a smart chatbot* that would help users to navigate the site and make the queries to the database and a *stand alone tool for personality identification*.

It is worth to shed some light on the methodology of personality identification.

First of all, there are two fundamental theories, which any personality study is based upon. Those are:

- Lexical hypothesis, which is defined by two postulates. The first states that those personality characteristics that are important to a group of people will eventually become a part of that group's language. The second follows the first, stating that more important personality characteristics are more likely to be encoded into language as a single word [1].
- Linguistic relativity, which is a hypothesis claiming that the structure of a language affects its speakers' world view or cognition, and thus person's perceptions are relative to their spoken or written language [2];

Although both of those are subject to critique, they are essential to the very concept of personality classification, as in their weakest forms the theories basically only state that the language and cognition are interconnected — the type of relation is what separates them and also what fuels the argument in the field.

There exist a few personality classification systems, such as big5 [3], 4 colors [4], Myers-Briggs type indicator [5]. The latter is considered to be the most applicable at the most, however, there is no unanimous decision on which systems describes existing types of personalities best. There is even no agreement on the fact that such system can be created in a way to describe all the possible variations in human behavior.

This question intrigued even ancient minds. For instance, Hippocrates reflected upon four types of temperament — sanguine, phlegmatic, choleric, and melancholic. Significant progress on the matter was made in 20th century with the development and availability of the statistical methods. Nonetheless, the polémique in the field is still vivid and no definite conclusion has been drawn.

In the scope of the current study, the idea of personality identification consists in generating a particular insight regarding person's personality from an example of their writing. This insight is intended to be represented by a personality class prediction, some recommendations on the communication with this person and any other possibly relevant information.

On the technical side, to fulfill the aforementioned objectives it was decided to use a vast functionality of IBM Cloud. Cloud development is arguably the most efficient approach, especially, when it comes to data science and digital transformation, that is available today.

Same as Google Cloud or Amazon Web Services, IBM Cloud offers to develop and deploy machine learning models, create data pipelines and data dashboards, organize clusters and deploy applications on them, etc. Besides that, IBM Cloud has a number of high level services, such as Personality Insight, Watson Assistant or Watson Discovery, capable to accelerate the development process. That is why it was only logical to benefit from all the offered possibilities.

2. Identification of personality profiles

2.1 Data

Keeping in mind the objective of the project, one should begin with its most important element — data.

As the goal is to produce an insight from an example of person's writing, the data one is going to be working with is text. There are no labels available at our disposal, which means that it is unknown which personality type each writing instance belongs to. Furthermore, it is unknown how many such types there might be. One may surely consider already existing studies on the matter, but none of them should be accepted in a belief kind of way.

Thus, a given task should be treated as an unsupervised learning one, namely clustering, due to the aim is to define groups with similar characteristics in an unlabeled dataset. Machine learning and AI-based algorithms need a lot of data to be trained on. Two sources that will provide this data are enterprise's clients mails database and an automated topic-based Twitter extraction.

First one speaks for itself — it is a set of plain text writings of company's clients. However, this database is not nearly enough to fulfill our needs, that is why it is reinforced with a second source. An example of script downloading the tweets of users related to the given topic is depicted in Fig. 2.1.

```
Searchig for users on 'Bukowski'
<<<>>>
Gathered 170 users and their timelines. Moving on.
Tweets of 159 users satisfy the given constraints. They are succesfully processed and saved.
<<<>>>
Total users gathered:  576
REQUESTS_MADE:  1058

Searchig for users on 'jazz legends'
<<<>>>
Gathered 15 users and their timelines. Moving on.
Tweets of 10 users satisfy the given constraints. They are succesfully processed and saved.
<<<>>>
Total users gathered:  586
REQUESTS_MADE:  2266
```

Figure 2.1 Twitter extraction script

It is worth to mention, what is being downloaded are not separate tweets, but whole timelines of a user, who was detected to tweet anything on the given topic. In such a way, a reasonable amount of one person's writing is obtained to be considered

representative. The timelines are processed using regular expressions to clean them from emojis, symbolic emojis and irrelevant symbols, such as "@" before names of other Twitter users.

The topics span from very general to specific, France or housing related ones. It was decided to consider profiles as diverse as possible to increase the generalization power of ML models, trained on this data.

Although one would like to determine personality type of a person given their writing example, text data is not what is used as a direct input to a model. Before feeding the data to the model it passes through a very important step of a pipeline — Personality Insight Service of IBM Cloud. Its principle of work will be described thoroughly in section 4, but it is necessary to be introduced at this moment for the sake of congruency.

This service allows to use a machine learning model that was trained on an enormous dataset (around million users for English) to produce a personality profile of a person from a text input. It has three groups of parameters:

- Big5
- Values
- Needs (divided into two groups for clarity purpose)

Those three groups in total comprise 22 following parameters:

Big5	Values	Needs I	Needs II
Openness	Conservation	Challenge	Liberty
Conscientiousness	Openness to change	Closeness	Love
Extraversion	Hedonism	Curiosity	Practicality
Agreeableness	Self-enhancement	Excitement	Self-expression
Emotional range	Self-transcendence	Harmony	Stability
-	-	Ideal	Structure

Table 2.1 — Personality traits

These parameters describe various traits of human personalities and they take their values in the [0,1] interval. A fragment of the obtained profiles dataset may be found in Fig.2.2.

	UID	Openness	Conscientiousness	Extraversion	Agreeableness	Emotional range	Challenge	Closeness	Curiosity	Excitement	...	Love
0	89168924	0.672358	0.515518	0.202560	0.388493	0.952090	0.449133	0.491036	0.373575	0.361041	...	0.669498
1	74580436	0.769577	0.442860	0.063138	0.152624	0.998766	0.792551	0.712586	0.726484	0.940915	...	0.964348
2	52536879	0.445128	0.776801	0.899640	0.881138	0.706536	0.857511	0.842577	0.751038	0.759625	...	0.899586
3	17243213	0.707812	0.725007	0.057476	0.296220	0.990815	0.551724	0.447475	0.519926	0.487398	...	0.542529
4	278662460	0.659059	0.788241	0.708605	0.238162	0.566861	0.686579	0.331890	0.248036	0.528839	...	0.247863

Figure 2.2 Personality profiles

It should be noted, that it would have been possible to train the algorithms directly on text. However, in this case one would need to adapt a more sophisticated preprocessing approach, use embeddings to transform each word into a vector of an arbitrary size and also define some functional layers, in case of ANN, or transformations, otherwise. Moreover, at the final stage it is quite likely that the data would be flattened into a vector of floats, again, of an arbitrary size. Such approach is very demanding in terms of both dataset size and training time.

On the other hand, a possibility to use Personality Insight, which produces a result that fits our purpose, and was trained with labels, obtained from countless social media surveys, is very promising.

All in all, this tool is too powerful to be neglected in the scope of the project.

2.2 Unsupervised learning and clustering

Following the description of the available data one concludes that the problem belongs to unsupervised learning domain. Unsupervised learning is a type of machine learning approach that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision. In contrast to supervised learning that usually makes use of human-labeled data, unsupervised learning, also known as self-organization allows for modeling of probability densities over inputs [6].

In most cases, one may address such problem with two generally applicable methods, which are clustering analysis and principal component analysis. The goal of the first one is to identify a given amount of groups of instances, which are called clusters, based on the underlying dependencies and variations in the data set, without any impact on the data itself. On the other hand, the objective of the second one is to identify these underlying dependencies themselves, possibly transforming data in such a way that they become clear and explicit. Due to the fact that after applying such methods, one obtains a data set of a lower dimensionality, which was exactly the reason for their application in the scope of the current study, detailed explanation of PCA and the description of its application will be given in the section 2.4.

There exists a wide variety of clustering methods. Mainly, it is due to the reason that beyond the basic approaches, each of them was carefully crafted to solve a specific problem, which is usually well defined by the properties of the data.

Next few methods were considered and applied in the frame of the research:

- K-means;
- Gaussian mixtures;
- DBSCAN (Density-based spatial clustering of applications with noise) [7];
- OPTICS (Ordering points to identify the clustering structure) [8].

Last two are density based methods: simply put they are looking for areas with high density of the instance occurrences and group them together. For instance, DBSCAN algorithm views clusters as subspaces of high density separated by subspaces of low density. Due to this rather direct view, clusters identified by DBSCAN can have any shape, as opposed to k-means which assumes that clusters are convex shaped. The central notion to this algorithm is the concept of core samples, which are high density area samples. Therefore, a cluster is defined as a set of core samples, each close to each other (measured by given distance measure) and a set of non-core samples that are close to a core sample.

While such approach may be quite efficient in the corresponding circumstances, none of the two density based methods produced any meaningful results in this case. Again, the main reason for that is the data we dispose. One may

observe eight arbitrary chosen data set parameters, plotted one against another in Fig. 2.3.

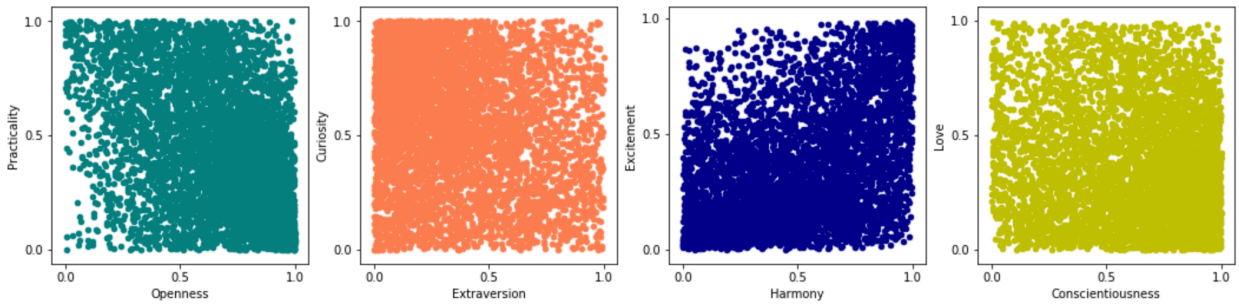

Figure 2.3 Pairwise plots of 8 arbitrary chosen parameters

As it might be observed, the data is incredibly dense and does not have any areas of low density. Thus, these methods are hardly applicable and mainly end up identifying one huge cluster, few incomparably small ones and a set of outliers. Results from one of the iterations are shown in Table 2.2 :

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Outliers
Share of samples	0.727	0.003	0.002	0.002	0.266

Table 2.2 — Cluster shares in DBSCAN example

That is why density based algorithms were not considered further for the current problem.

K-means and Gaussian mixtures (GM) produced more meaningful results and were considered to be quite fitting the problem, so it is necessary to give more details on these algorithms.

The concept of K-means method first appeared in 1957 and was proposed by Hugo Steinhaus[9], however, this particular name was given to the algorithm later. Given N — number of samples, K — desired number of clusters, the main idea of the approach is to apply an iterative technique to find the best partition of data set into K groups $S = \{S_1, S_2, \ldots S_K\}$:

1. Initialize K means (cluster cores or centroids) in an arbitrary way;
2. Assign each of the N observations to the closest core;

3. Update the cores: $m_i^{(t)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$

4. If the assignments does not change — the algorithm has converged, otherwise — return to 2.

Thus, the problem may be reformulated in a more concise form as:

$argmin_S \sum_{i=1}^{k} \sum_{x \in S_i} d(x, m_i)$, where $d(x, y)$ is a chosen distance. Euclidian distance is the one most commonly used, or used by default, but one may choose any other distance depending on the encountered task.

All in all, one may state that this is a fairly straightforward algorithm; it is often used as a baseline approach. Despite that, it is very powerful, having the only assumption on the convexity of clusters.

A Gaussian mixture model is a probabilistic model that assumes all the data points in the dataset are drawn from a finite number of Gaussian distributions with unknown parameters, that are to be found. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians. This kind of model and the problem of mixture decomposition that is the identification of its constituent components and the parameters thereof, has been cited in the literature as far back as 1846 [10], but common reference chiefly made to the work of Karl Pearson (1894).

Mathematically, in its most common form a model may be described as follows:

- K — number of cluster, or mixture components, to find;
- N — number of observations in the dataset;
- μ — a K-dimensional vector of means of each hidden distribution;
- σ — a K-dimensional vector of standard deviations of each hidden distribution;
- ϕ — a K-dimensional vector of weights of each hidden distribution;
- $F(x|\mu, \sigma)$ — probability distribution of an observation, parametrized on μ and σ, where x is one observation in the dataset X of size N.

Then the model of the probability distribution of the data may be written in the next form: $M = \phi F(X, \mu, \sigma) = \sum_{i=1}^{K} \phi_i F(X, \mu_i, \sigma_i)$. In the case of gaussian mixture, $F(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$ is of course a gaussian, but the approach might be extended to any distribution type, depending on the assumption regarding the data. Then, vectors μ, σ may be replaced by a single parametrized vector θ that would hold the parameters of the corresponding distribution.

The model is optimized using the expectation maximization algorithm. This is an iterative approach, aiming to find a local maximum of the likelihood function by alternating the values of weights and parameters vector. Expectation maximization is seemingly the most popular technique used to determine the parameters of a mixture with a given number of components. One example may be found and studied in detail in Dempster et al (1977) [11].

This idea to try this method came up after the examination of the following pairwise plot (Fig. 2.4) with respect to the class, identified by K-means.

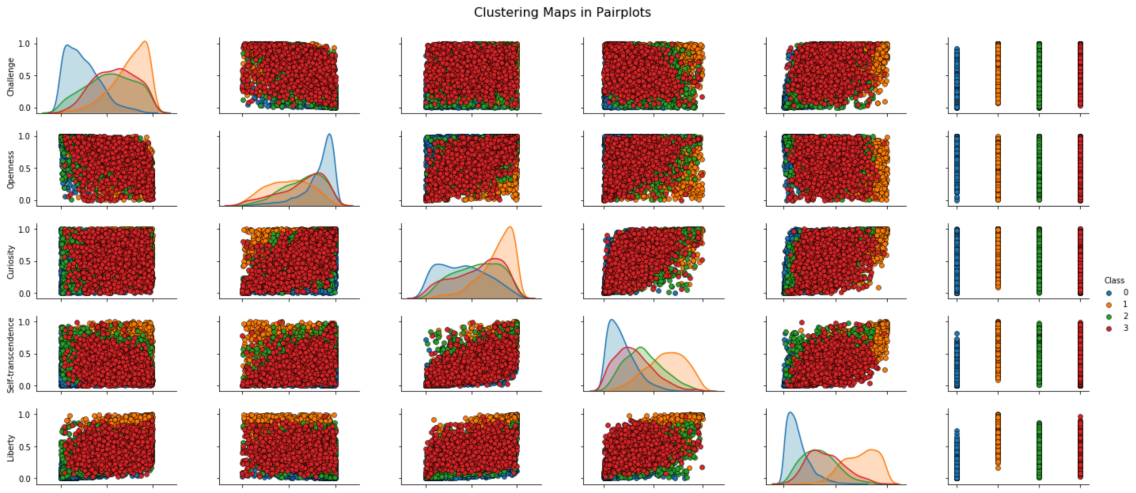

Figure 2.4 Pairwise plots with respect to class

It may be argued that the kernel plots on the diagonal may not all be called as such, containing gaussians, however, there is certainly a tendency towards the bell curve of the normal distribution. An assumption on the normality of the data distribution is of course a strong one, but no indication was found that such data

should belong to a different kind of distribution, and while the aforementioned one is considered to be no less than normal, thus, the most commonly encountered, it was only logical to at least give it a try.

It is essential to remark, that both "K-means" and "Gaussian mixtures" algorithms operate with distance. The notion of distance and its definition is crucial for an unsupervised learning problem, so the findings regarding it will be presented in the following section.

2.3 Choice of metric

The choice of metric is, certainly, one of the key problems to solve in the scope of a clustering problem. By definition, a metric or distance function is a function that defines a distance between each pair of point elements of a set. Thus, it is irrefutably imperative to define and use it properly in such context.

It is a point of even greater concern when dealing with high-dimensional data. Recent research shows that in high dimensional space, the concepts of distance, proximity or nearest neighbor may not even be qualitatively meaningful. There are studies, which indicate and explain the inefficiency of Euclidian norm, and usage of the extension of L_k norm to the fractional values of k [12]. These results were taken into account and multiple variations of L_k norm were tried.

Besides that, a special kind of metric, which is thought to be more appropriate to use when working in higher dimensions, was studied and tried on practice. This metric is Mahalanobis distance, introduced by P. C. Mahalanobis in 1936.

Given $P \in \mathbb{R}^N$, a point, and D, a distribution in the same N-dimensional space, it is a multi-dimensional generalization of the idea of measuring how many standard deviations away P is from the mean of D.

This distance is zero if P is at the mean of D, and grows as P moves away from the mean along each principal component axis. If each of these axes is re-scaled to have unit variance, then the Mahalanobis distance corresponds to standard Euclidean distance in the transformed space [13].

Let D have mean $\bar{\mu} = (\mu_1, \mu_2, \ldots, \mu_N)$ and covariance matrix S, considering a point $\bar{x} = (x_1, x_2, \ldots, x_N)$, $\bar{x} \in \mathbb{R}^N$,

Mahalanobis distance is then defined as $d_M(\bar{x}) = \sqrt{(\bar{x} - \bar{\mu})^T S^{-1}(\bar{x} - \bar{\mu})}$.

It may as well be defined for two random vectors $\bar{x}$ and $\bar{y}$ of the same distribution: $d_M(\bar{x}, \bar{y}) = \sqrt{(\bar{x} - \bar{y})^T S^{-1}(\bar{x} - \bar{y})}$.

It is worth to remark, that if covariance matrix S is the identity matrix, the above definite reduces to Euclidian distance. Thus, one may view it as a generalization of Euclidian distance. In the scope of the studied problem, it corresponds to the ability to take into account the "spread" of the cluster, not only proximity of the point to its core.

As it was mentioned before, Mahalanobis distance is especially applicable in high-dimensional problems. The inclusion of covariance matrix, obviously, allows to consider more information regarding point's relative position in a given space. Indeed, its application on clustering of the profiles data set has proved to produce more homogeneous and balanced (in terms of the number of instances) clusters, notably, when used in Gaussian mixtures model (Table 2.3).

Distance / Cluster	0	1	2	3	4	5
L_1	0.988	0.001	0.001	0.005	0.0	0.004
D_M	0.708	0.074	0.074	0.016	0.04	0.088

Table 2.3 — Shares of clusters for L_1 and D_M in GM clustering

However, these properties tend to disappear when working with a projection of the dataset in 2D. This is also logical, since the additional information gained from covariance matrix is diminishing with the decrease of dimension.

The choice of metric impacts not only the functioning of the clustering algorithms, but also the algorithms of evaluation of obtained partition, which are described in section 2.5. Two principal measurements which are used to quantify the efficiency of clustering may be intuitively compared to homogeneity (cohesion) and uniqueness (or separation) of a given cluster. In this context, one or another distance puts more emphasis on the first or the second, depending on its properties.

Finally, there is of course numerous other ways to define a distance which may all be proven logical and fitting the problem. The question of the choice of a right one may even be called arbitrary, as there is always a trade-off of some kind in place.

It is doubtful, that a particular metric may be called the best for given endeavor, as it is doubtful that there is one that shows a clear edge over other ones universally, during all the stages of the project. This topic is subject to debates and argument in scientific community, meaning no definitive answer is available at the moment. Thus, in case of the continuation of the project, it is recommended that this issue is reevaluated and thought through.

2.4 Dimensionality reduction

This section describes the dimensionality reduction techniques that were studied and applied in the current research. Their purpose is to express the original data in such a way that maximum of information is gained from the data, while it is also projected to a lower dimension.

It is achieved by making latent variables, which are called factors, appear from the initial set of variables. Principal component analysis (PCA) is a process of finding these underlying factors. PCA was invented in 1901 by Karl Pearson and is defined as an orthogonal linear transformation that projects the data to a new coordinate system, such that the greatest variance of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on [14].

Considering a $n \times p$ data matrix X, mathematically, the transformation is defined by a set of m p-dimensional vectors of weights $w_k = (w_1, w_2, \ldots, w_p)_{(k)}$ that map each row vector of X (one observation) into a vector $y_i = (y_1, y_2, \ldots, y_m)_{(i)}$ of a new space with $m < p$, preferably. Then, $y_i^k = w_k \cdot x_i$, for $k = 1,\ldots, m$ and $i = 1,\ldots, n$.

To maximize the variance, according to the definition above, w_1 must satisfy $w_1 = argmax_{||w||=1} \sum_i (y_i^1)^2$ or equivalently in the matrix form:

$$w_1 = argmax_{||w||=1}\{||Xw||^2\} = argmax_{||w||=1}\{w^T X^T X w\},$$

Since w is defined as a unit vector, one may also rewrite the above expression as:

$w_1 = argmax\{\frac{w^T X^T X w}{w^T w}\}$, which is a Rayleigh quotient. That gives one a possibility to use a known result for any positive semidefinite matrix, which $X^T X$ is, to conclude that the quotient maximum value is achieved when w is the eigenvector that corresponds to the largest eigenvalue of the matrix.

The reasoning is the same for every value of k, except for matrix X should be updated in the following way:

$$\hat{X}_k = X - \sum_{j=1}^{k-1} X w_j w_j^T.$$

There are other ways to achieve the same result, but it is certainly not the topic of the study.

While this algorithm is fairly robust, it is also pretty simple and so it is considered a basic version of another technique — factor analysis (FA). There has been significant controversy in the field over differences between the two techniques, but it is thought, that factor analysis is clearly designed with the objective to identify certain unobservable factors from the observed variables, whereas PCA does not directly address this objective; at best, PCA provides an approximation to the required factors [14].

Considering same dataset X with p parameters with corresponding means $\mu_1, \ldots, \mu_p$; a vector of unknown constants $l_1, \ldots, l_m$ and m factors $F_1, \ldots, F_m$, one has that given the original variables, reduced by the values of their means, it should be possible to represent them as linear combinations of factors, as following:

$x_k - \mu_k = l_{k1} F_1 + \ldots + l_{km} F_m + \varepsilon_k, \quad k = 1,\ldots p$; where ε_k are unobserved stochastic errors with zero means and finite variances.

So, to obtain representations of original data in the form of factors, one should solve $X - \mu = LF + \varepsilon$ under the following assumptions regarding F:

1. F and ε are independent;
2. $\mathbb{E}(F) = 0$
3. $Cov(F) = I$

L is called a loading matrix and it is unknown as well. However, it is known that $Cov(X-\mu) = Cov(LF+\varepsilon)$. Let it be noted by Σ, then, $\Sigma = LCov(F)L^T + Cov(\varepsilon)$. Or, considering the assumptions and denoting $Cov(\varepsilon)$ as Φ, one obtains that $\Sigma = LL^T + \Phi$.

From the above comes the difference with PCA: from the point of view of factor analysis, the eigenvalues of PCA are inflated component loadings, i.e., contaminated with error variance [15].

Principal component analysis is widely used as a base method for factor extraction. Then, various techniques, such as maximum likelihood, may be applied to find the "true", unbiased loading matrices or to evaluate the quality of the representation, to test the statistical significance of factor loadings [15].

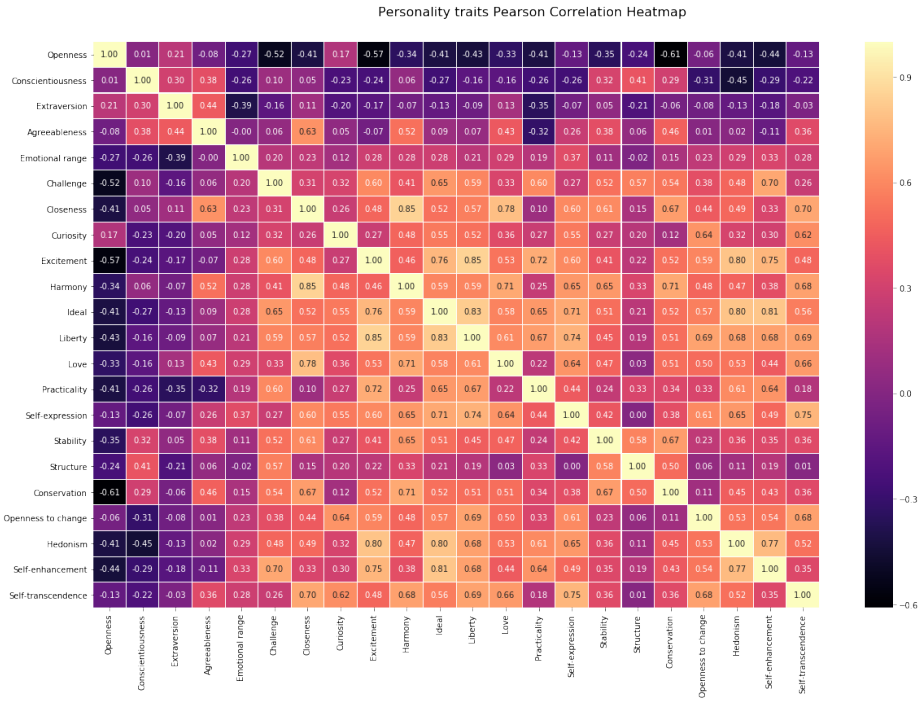

Figure 2.5 Pearson correlation matrix of personality traits

To depict the reason for conducting dimensionality reduction more clearly, it is suggested to examine figures 2.5 and 2.6, where correlation matrices of original data set and of the transformed one are offered.

It may be observed, that the newly found factor are indeed centered around zero and have no correlation between them.

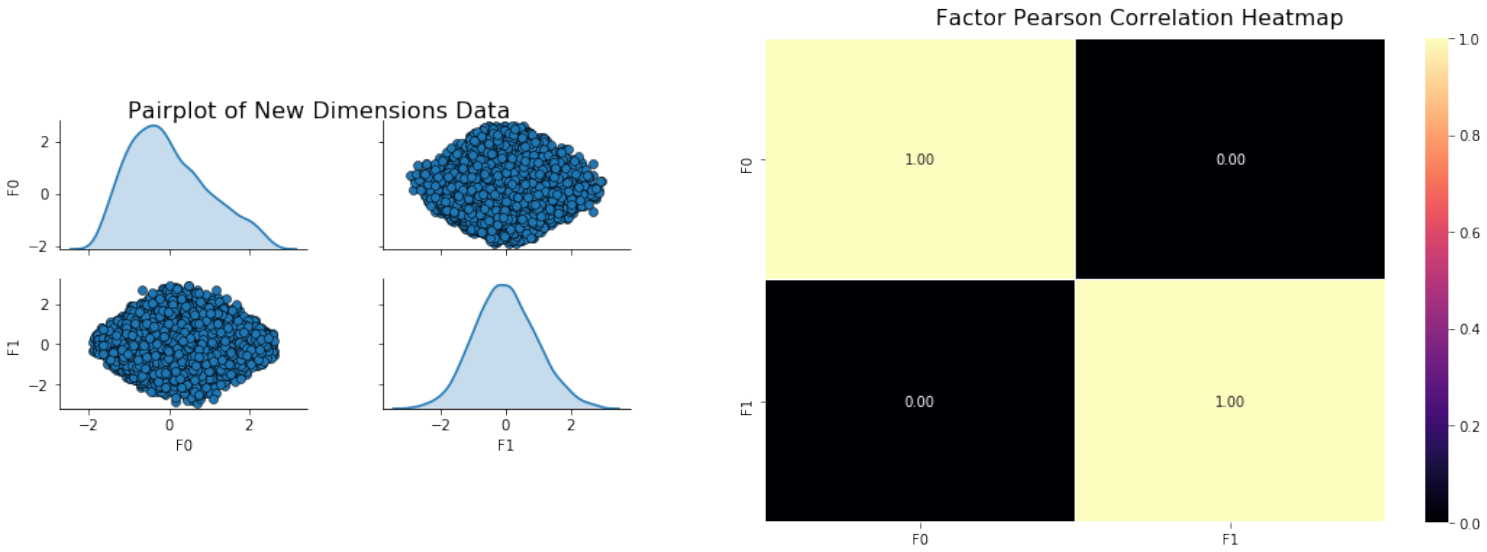

Figure 2.6 Pearson correlation matrix of detected factors (2D)

These two approaches to dimensionality reduction may be considered conventional. While they can be very powerful and efficient, it was decided to test a rather unconventional approach to this part of the problem. It is built around autoencoder type of machine learning model.

An autoencoder is a neural network trained to attempt to copy its input to its output. Internally, it has a hidden layer h that describes a code used to represent the input. The network may be viewed as consisting of two parts: an encoder function $h = f(x)$ and a decoder that produces a reconstruction $r = g(h)$. If an autoencoder succeeds in simply learning to set $g(f(x)) = x$ everywhere, then it is not especially useful. Instead, autoencoders are designed to be unable to learn to copy perfectly. Usually they are restricted in ways that allow them to copy only approximately, and to copy only input that resembles the training data. Because the model is forced to prioritize which aspects of the input should be copied, it often learns useful properties of the data [17].

The idea of autoencoders has been part of the historical landscape of neural networks for decades, with works, like Ballard [18], dating back to 1987. Recently, theoretical connections between autoencoders and latent variable models have brought autoencoders to the forefront of generative modeling. Autoencoders may be thought of as being a special case of feedforward networks, and may be trained with all of the same techniques, typically minibatch gradient descent following gradients computed by back-propagation [17].

In the studied case, the dimension of the encoding layer h is smaller than that of the input, due to the goal of the endeavor is reduction of dimensionality. In such a case, this model is said to be called undercomplete autoencoder. It would have the following architecture:

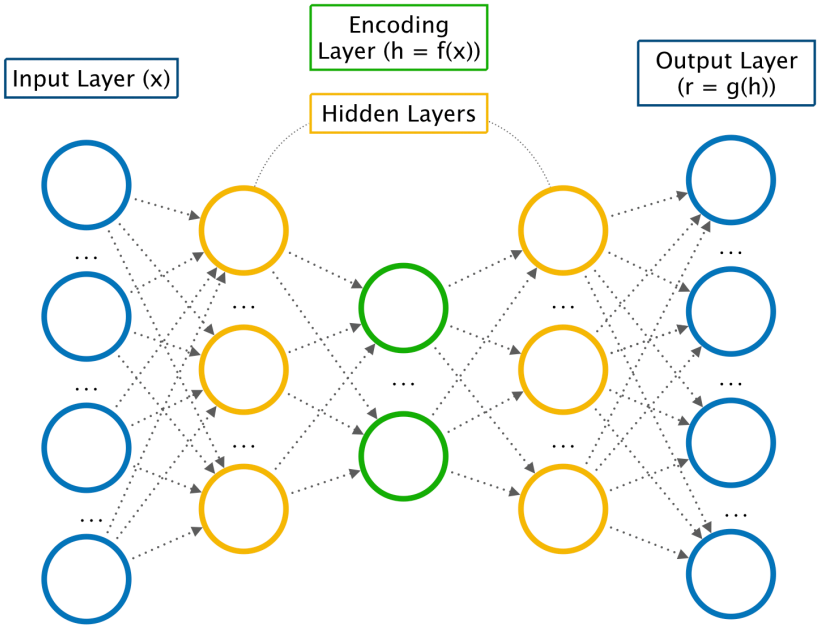

Figure 2.7Architecture of undercomplete autoencoder

Learning an undercomplete representation forces the autoencoder to capture the most salient features of the training data, as it is forced to compress the information of higher dimension into a lower dimension.

The learning process itself is described simply as minimization of a loss function $L(x, g(f(x))$ of a chosen kind. The criterion to satisfy is that it should penalize $g(f(x))$ for being dissimilar from x.

Notably, when the decoder is linear and L is the mean squared error, an undercomplete autoencoder learns to span the same subspace as PCA, described earlier. In this case, an autoencoder, trained to perform the copying task, has learned the principal subspace of the training data as a side-effect [17].

That leads one to a thought that, potentially, autoencoder with nonlinear encoding and decoding functions, $f(x)$ and $g(h)$, may learn a more powerful nonlinear representation of PCA. This is particularly intriguing in the studied case, due to, having examined Fig. 2.5 in detail, one may notice that measures of linear correlation

are not too high for the profile traits data. Moreover, one may consider deeper autoencoder, where $h = f(x)$ is, in fact, $f(x) = f(f_1(f_2(x)))$, for instance; correspondingly, r would also be $r(r_1(r_2(h)))$.

Unfortunately, when autoencoder is given too much capacity, it would learn the most intricate detail of the given data, however it would perform only the copying task without extracting useful information about the distribution of the data. Thus, one should keep that in mind when training such model.

To sum up, all three suggested approaches may help one to achieve the goal of reducing the dimensionality of a dataset, however, they perform the task it in slightly different ways. In section 2.6 their comparison is given. The chosen approach is explained in terms of its application on practice as well as it is explained why it was decided to choose it.

2.5 Evaluation of the partition

Unsupervised learning presents a particular difficulty — absence of the ground truth, which means that there is no direct way to evaluate the obtained result and compare different models. Thus, one has to define a set of measurements that will serve this purpose according to the objective.

In this section, these measurements will be presented.

First of all, it was chosen to consider measuring cluster tendency, which is a way to tell how well the dataset may be clustered. For that cause Hopkins statistic was applied [19]. It belongs to the family of sparse sampling tests, due to only a sample X from the dataset is considered. To counteract with X, a random sample Y of uniform distribution is generated. The statistic is calculated in the following way:

$H = \frac{\sum_{i=1}^{m} u_i^d}{\sum_{i=1}^{m} u_i^d + \sum_{i=1}^{m} w_i^d}$, where d is the dimension of the data, m is the size of the sample to draw from the dataset, u_i is the distance between $y_i \in Y$ and its nearer neighbor in X, or the "outer distance", w_i — distance from $x_i \in X$ to its nearest neighbor in X, or the "inner distance".

It is said that the index approaches 1 for the data with high cluster tendency. Launching this procedure multiple times for additional certainty for raw data, data reduced with FA and for data reduced with autoencoder one obtains:

```
for i in range(10):
    print("Raw: {:.4f}; FA: {:.4f}; AutoEnc.: {:.4f}".format(Hopkins_statistic(profiles_raw, Lk_norm, m=100),
                                                             Hopkins_statistic(profiles_FA, Lk_norm, m=100),
                                                             Hopkins_statistic(profiles, Lk_norm, m=100)))
```

```
Raw: 0.9929; FA: 0.6402; AutoEnc.: 0.9989
Raw: 0.9938; FA: 0.4060; AutoEnc.: 0.9986
Raw: 0.9996; FA: 0.3972; AutoEnc.: 0.9991
Raw: 0.9995; FA: 0.2796; AutoEnc.: 0.9993
Raw: 0.9959; FA: 0.3310; AutoEnc.: 0.9989
Raw: 0.9993; FA: 0.4123; AutoEnc.: 0.9992
Raw: 0.9999; FA: 0.3383; AutoEnc.: 0.9992
Raw: 0.9960; FA: 0.2587; AutoEnc.: 0.9986
Raw: 0.9999; FA: 0.3954; AutoEnc.: 0.9991
Raw: 0.9983; FA: 0.5814; AutoEnc.: 0.9992
```

Figure 2.8 Hopkins statistic calculation

Notably, one may observe the clustering tendency is almost as high as it could be for raw and autoencoder transformed data, while it is significantly lower for FA transformed data. That might be explained by the very goal of FA, that aims to find the best axis to spread the data along them, which means that it may very well approach uniform distribution in the reduced domain. At this point, no conclusions will be drawn solely on the result of this measurement, nonetheless, one should keep it in mind.

Hopkins statistic is a way to measure, so to say, a priori cluster tendency. Next two measures — Davies-Bouldin index and Silhouette coefficient — should be applied after clustering is performed. They aim to evaluate the quality of obtained partition.

The Davies–Bouldin index (DBI) was introduced by David L. Davies and Donald W. Bouldin in 1979. This is an internal evaluation scheme, where the validation of how well the clustering has been done is made using quantities and features inherent to the dataset [20].

There are few steps to calculate the index. Let C_i be a cluster of size T_i with the centroid A_i that contains points X_j, $j = 1,\ldots, T_i$. First, one should measure the scatter for each cluster:

$$S_i = \frac{1}{T_i}\sum_{j=1}^{T_i} d(X_j, A_i), \text{ where } d(\cdot,\cdot) \text{ is a chosen distance.}$$

We, then, define an inter cluster distance $M_{i,j}$ simple as a distance between their corresponding centroids: $M_{i,j} = ||A_i - A_j||$.

Now, let $R_{i,j}$ be a measure of how good the clustering scheme for the two particular clusters is. To fulfill its objective, the measure should consider $M_{i,j}$, the separation between the ith and the jth cluster, which ideally has to be as large as possible, and S_i, S_j, the within cluster scatter for clusters i and j, which should be as low as possible. So, it is defined as $R_{i,j} = \frac{S_i + S_j}{M_{i,j}}$. Then, it has following properties:

1. $R_{i,j} \geq 0$
2. $R_{i,j} = R_{j,i}$
3. The lower the value of $R_{i,j}$, the better the separation of the clusters and the more homogeneous each of them is.

Finally, Davies-Bouldin (DB) index is then computed as:

$DB = \frac{1}{K} \sum_{i=1}^{K} \max_{i \neq j} R_{i,j}$, where K — is the number of found clusters.

It is worth to note, that choosing maximum value of $R_{i,j}$ is fairly strict, as only worst-case scenario is taken into account, but one can tailor the computation, for instance, by considering average value instead of maximum, in case it is more fitting the problem.

Second measure is Silhouette coefficient. It refers to a method of interpretation and validation of consistency within clusters of data. The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from -1 to $+1$, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. If many points have a low or negative value, then the clustering configuration may have too many or too few clusters [21].

As well as Davies-Bouldin index it may be calculated using various distance metrics. However, it has an advantage of the possibility for clear and descriptive visualization.

Again, given an established partition with K clusters C_i, $i = 1,\ldots,K$, first step is computation of a_j, which may be interpreted as a measure of how well a point is assigned to its cluster, as $a_j = \frac{1}{T_i - 1} \sum_{k=1,k\neq j}^{T_i} d(x_j, x_k)$, where T_i is the magnitude of the cluster, as before.

Then, for each point in a given cluster C_i we compute the mean distance from this point to all the points in a different cluster C_k as a measure of similarity to the cluster that it does not belong to. As in Davies-Bouldin index only worst case is of interest, so one proceeds by computing $b_j = \min_{k\neq i} \frac{1}{T_k} \sum_{x_u \in C_k} d(x_j, x_u)$.

The cluster with the smallest mean dissimilarity is said to be the "neighboring cluster" of the point x_j because it is next best fit cluster for it.

Silhouette value is then defined as $s_j = \frac{b_j - a_j}{\max\{a_j, b_j\}}$, if $T_i > 1$, and $s_j = 0$, if $T_i = 1$.

Interpreting this formula, one finds out that it is desirable that a_j is as small as possible to obtain a good partition. That finds its reflection on the definition of a_j as a mean distance to all the points in its cluster, which stands for cohesion or homogeneity of the cluster. On the other hand, the higher the value of b_j — the higher is the degree of separation between the point's cluster and its nearest neighbor.

Silhouette value is the mean $s = \frac{1}{N} \sum_{i=1}^{N} s_i$ over all points of a dataset, a measure of how tightly grouped all the points are. In the same way silhouette coefficient may be computed for each particular cluster.

It is worth to remark, that contrary to Davies-Bouldin index, Silhouette does not at all involve centroids of the clusters. Essentially measuring same components — intra-cluster similarity (cohesion) and inter-cluster similarity (separation) — quite different approaches are used.

While first measures the distances between each cluster core and all the points of the cluster as a measure of cohesion, second one measures the distances of each

point to all kindred. Same holds for the approach to computation of separation. One also notes significant difference in computational difficulty of two algorithms.

There are many possible ways to interpret the logic behind the two approaches, especially, when applying the algorithms on a real-world data, but, what is important to remark here — is that, together with cluster tendency estimation, a set of measures, capable of giving a well supported estimation of the partition from multiple perspectives, was defined. All aforementioned algorithms were implemented and used as evaluation criteria for all the studied models and data reduction approaches.

2.6 Results

The final section of this chapter presents the obtained results, conclusions and ideas for the future development of the "Personality Identification" project. All the algorithms and concepts, discussed in sections 2.2-2.5, were implemented and tested. The final solution is comprised of dimensionality reduction algorithm, clustering algorithm and the chosen metric. A number of such combinations was studied and evaluated using the defined measures.

To optimize meta-parameters of each algorithm and to choose the best one, grid search technique was used. For instance, results of grid search for best new dimension and number of clusters for combinations "FA + kmeans" and "FA + GM" may be inspected in Fig. 2.9 and Fig. 2.10 correspondingly.

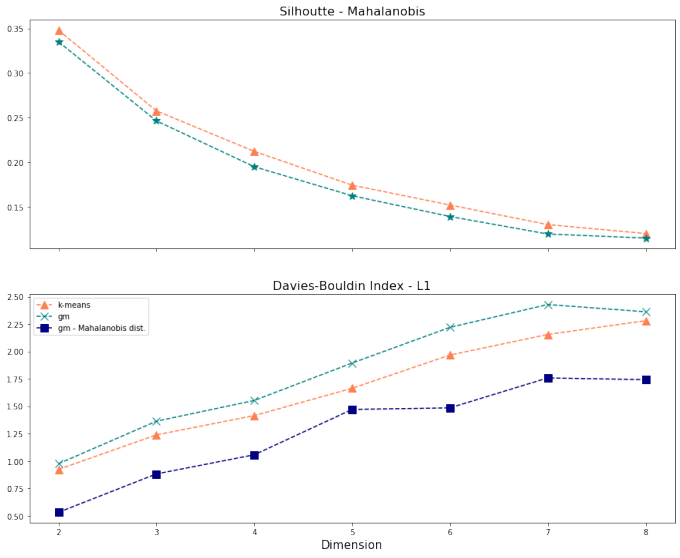

Figure 2.9 Grid search of dimension (FA)

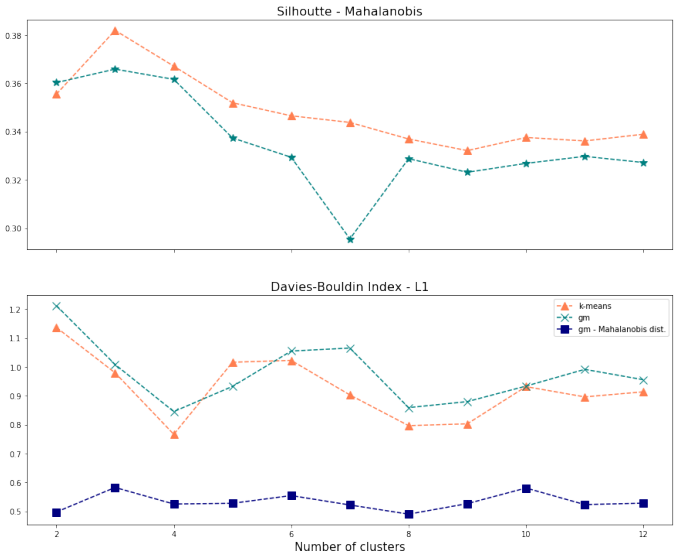

Figure 2.10 Grid search of number of clusters (FA)

Same experiment was conducted using autoencoder instead of FA. These results may be observed in Fig. 2.11, 2.12.

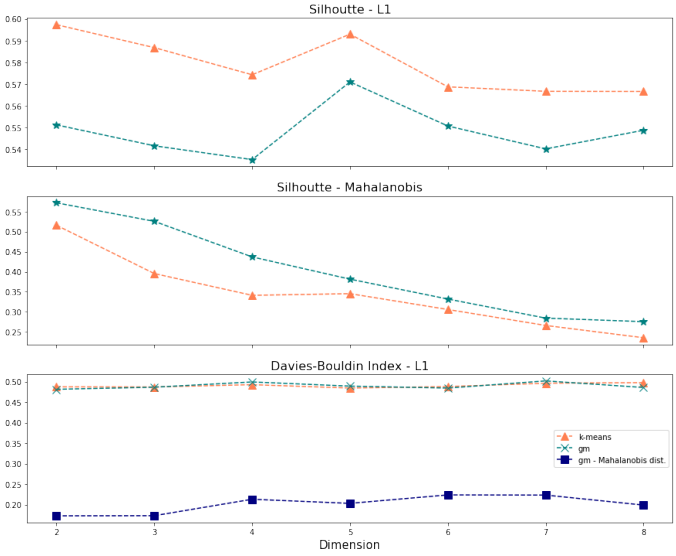

Figure 2.11 Grid search of dimension (Autoencoder)

For both FA and autoencoder reduced data, best Silhouette score and DBI were obtained in the 2D case. However, there is also a spike for 5D in the case of autoencoder. Next charts depict results for these two cases with varying number of searched clusters.

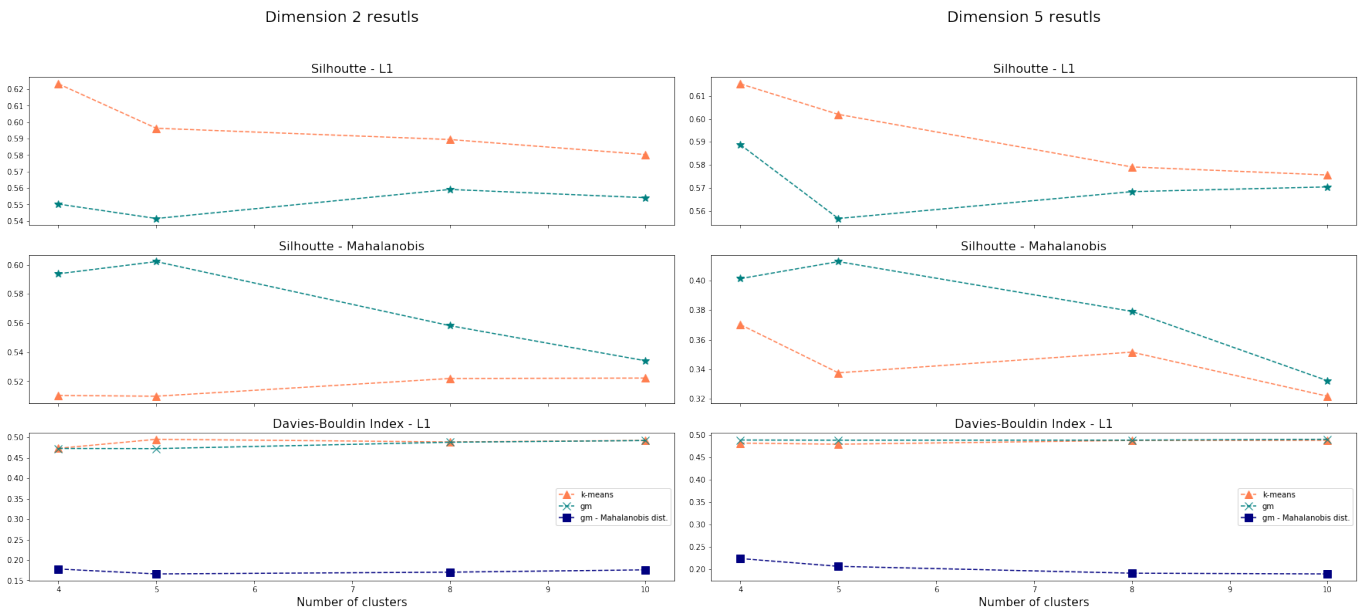

Figure 2.12 Clustering evaluation in 2D and 5D (Autoencoder)

Based on the observed results, one may make following conclusions:

• Data reduction into 2D clearly shows the most promising results;

• Incorporation of Mahalanobis distance does not bear any fruit. One assumes that it is due to the dimensionality reduction procedure, which neutralizes the additional information gain from covariance matrix;

• Gaussian mixtures in almost all the cases scores slightly worse than k-means for FA reduced data, but the gap increases when models work with autoencoder reduced data;

• Davies-Bouldin index shows extremely low variation in the case of autoencoder reduced data, however, it reaches fairly low value, which is desirable;

• Comparison of the results for different number of clusters is rather unclear: the change in the scores is not as noticeable as for different dimensions; significant fluctuations in Fig. 2.10 defy any assumptions regarding that. The difference up to a value of 0.05 may be easily explained by the random factors, wchih influence the training processes of models (such as initial centroids coordinates, order and content of the batches, etc.). Thus, it might be possible, that any number of clusters can be justified, especially, if sufficient theoretical argumentation from field expert is presented.

All in all, based on everything mentioned above, it was decided to choose autoencoder as dimensionality reduction technique for showing higher scores, and k-means as a clustering algorithm following Occam's razor principle. Besides, it is generally considered that the main disadvantage of the algorithm is its inefficiency in higher dimension, which is not in the reduced dimension space. L_1 distance was chosen both for the clustering algorithm and for its evaluation as no reason to choose any alternative was available.

After the final approach was chosen, a number of ways to further improve its performance were considered. A possible way for such improvement, found in *"Unsupervised Deep Embedding for Clustering Analysis"*[22] was studied and implemented. The key idea in the paper is to minimize Kullback-Leibler divergence between existing distribution of probabilities that an instance belongs to a given cluster and a target distribution, which should be given to a model. However, it was not specified how to define the target distribution, especially in the real-case scenario instead of MNIST data. Unfortunately, this approach was not capable of increasing clustering scores.

Other ideas for possible improvements revolved around optimization of the partition found with k-means. They included application of Hopfield networks [23] or genetic algorithms [24]. Theoretical basis for theses algorithms was briefly studied, as well as some available implementations, however, these ideas did not advance beyond the conceptualization stage in the scope of the given project. Nonetheless, they might be of interest in case of continuation of the work on this endeavor.

Otherwise, another point to mention is visualization and presentation of the obtained results. As a model's prediction, one obtains a vector of cluster assignments for each point of the dataset. It is concatenated with the initial dataset (22D) to perform an analysis of the proposed solution. Final model's evaluation metrics are given in Table 2.4. Silhouette scores for a sample drawn from each cluster is shown in Fig. 2.13.

Metric	Silhouette score	Sil. cluster 0	Sil. cluster 1	Sil. cluster 2	Sil. cluster 3	DBI
Score	0.681	0.568	0.856	0.75	0.557	0.428

Table 2.4. Evaluation metrics of the partition, produced by the final model

Multiple variations of the same model, which were obtained during training optimization, produced similar results, with Silhouette score around 0.63 and DBI ≈ 0.45.

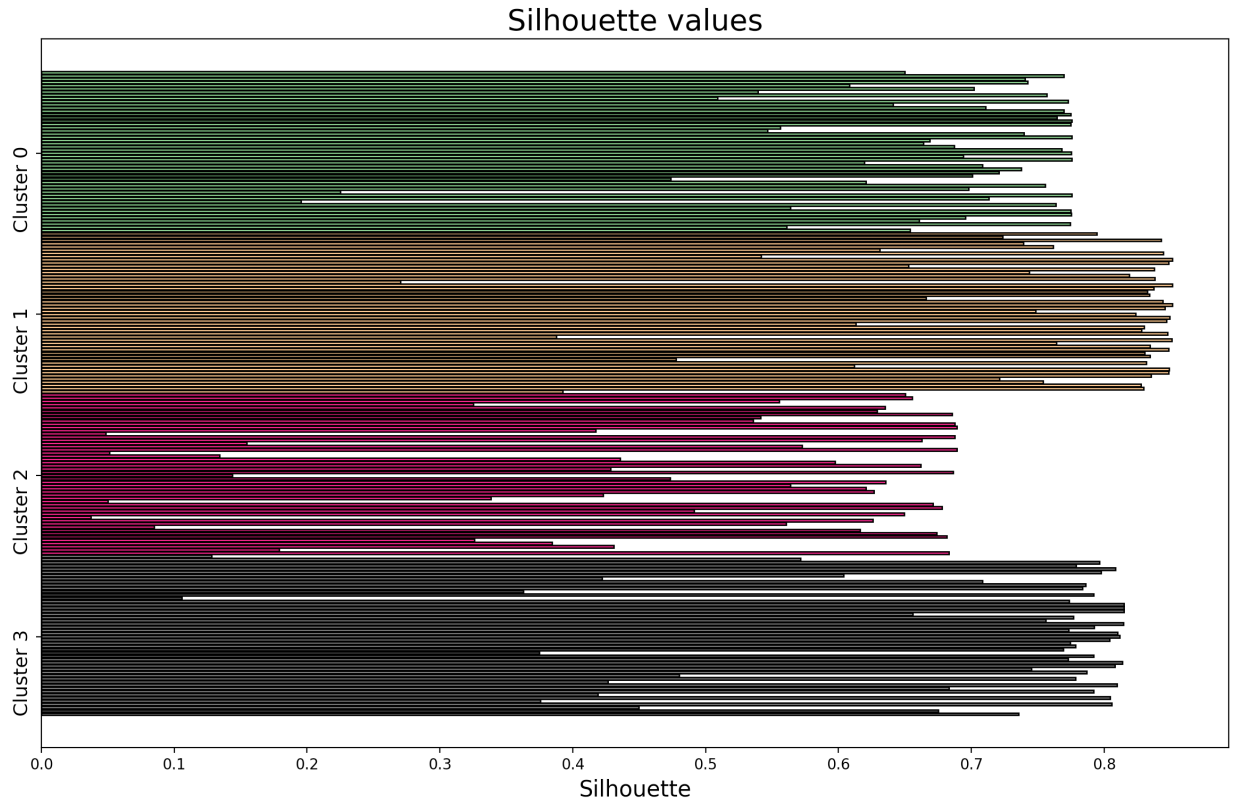

Figure 2.13 Silhouette scores for sample ($n = 100$) drawn from each cluster

It may be observed, that these particular instances are well classified, on average. There are some drops amongst cluster 0 and cluster 2 instances, but average scores are higher than 0.5 for each of the groups.

The current partition has not only scored well on evaluation metrics, but also produced a very well structured 2D space, with balanced factors (Fig. 2.14).

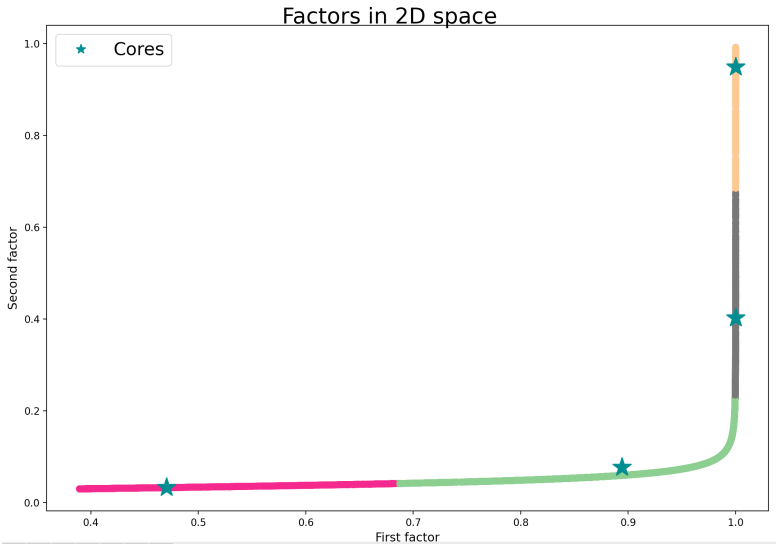

Figure 2.14 Obtained factors in 2D space

As it may be observed, the found factors form a rather generic curve, with cluster centroids occupying its different sides. One may expect some confusion and lower silhouette scores for the points, situated on the edges of each color zone.

From the point of the generation of the insight, following visualization contains the most valuable information. It is depicted in Fig. 2.15 for all 4 clusters. This graph is a polar plot of trait values of the centroids of each cluster. In other words, it represents a typical profile of a person, belonging to the group, according of course to the partition obtained by the model, described above. Trait names only shown for one cluster in each column to avoid overcrowding the picture. To fulfill the objective to the fullest, these typical profiles should be studied and interpreted, preferably with participation of a field expert (psychologist, market specialist, etc.). Although such field expert was not available, such analysis was conducted and clusters were given concise interpretations.

Before describing each cluster, it is worth to shed some light on the meaning of each traits group. Big Five is one of the most studied of the personality models that were developed by psychologists [25]. It is the most commonly used personality model to describe how a person engages with the world in general. Needs are important aspects of human behavior. Research literature suggests several types of human needs are universal and directly influence consumer behavior. The twelve categories of needs that are reported by the service are described in marketing literature as desires that people hope to fulfill when they consider a product or

service. Values convey what is most important to an individual. They are "desirable, trans-situational goals, varying in importance, serving as guiding principles in people's lives" [26]. They are beliefs, motivations, the very reasoning and guidance systems of a person.

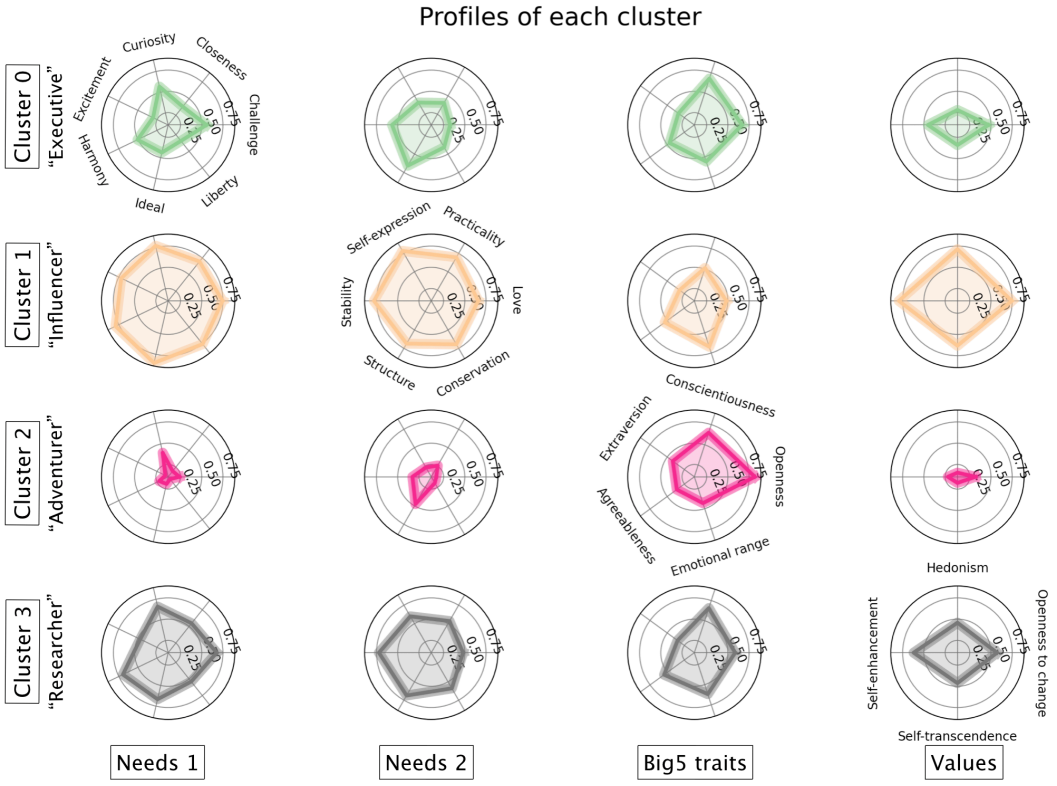

Figure 2.15 Cluster profiles

Cluster 0 was called "Executive". Its highest scores are conscientiousness, structure, curiosity and challenge. The "Values" are not very well defined, the highest being self-enhancement and openness to change, approaching 0.5. "Needs", such as Liberty, Ideal or Self-expression are very low. So, the ones belonging to the cluster are disciplined, diligent people aiming to succeed and understand what they are doing. However, they seem to ignore the abstract individual concepts such as a liberty, mostly functioning in the material part of the world.

On the contrary, cluster 1 was called "Influencer". These are incredibly needy people, with their scores in "Needs" passing 0.75 border in almost all the cases. Their most significant big5 trait is Emotional range, high values of which correspond to the propensity to express emotional behavior. This group also has the highest scores of "Values", with Hedonism and self-enhancement reaching 0.75 and 0.8 correspondingly. In combination with high scores in self-expression and Ideal, it may

be concluded that these people tend to express themselves and to spread their influence and ideas on others.

Cluster 2 was given a name "Adventurer". This cluster differs a lot from all the others. Its most prominent feature is the highest among all clusters score (0.82) on Openness. It also has the highest score on Extraversion (0.42) and Conscientiousness value slightly smaller than that of centroid of cluster 0. However, the representatives of this group almost entirely ignore "Needs" and "Values", having only Curiosity, Structure and Openness to change as the only traits approaching 0.5 threshold. Thus, one may assume, that these people are not the typical consumers. One may as well speculate that they may not be influenced easily and may manifest unconventional or new ideas and beliefs.

Finally, last cluster was named "Researcher". This cluster is somewhat alike to cluster 0 in terms of Big5 traits, however, unlike "Executive", this cluster shows a rather more developed "Needs" and "Values" systems. Besides Structure and Curiosity "Researchers" seem to need Stability, Harmony and demonstrate significant desire for challenge (0.68). They also have the most balanced values composition, with all the scores laying in range {0.45,0.65}, highest being Self-enhancement.

This interpretation of the obtained clustering is not aimed to be perceived as a definite description of possible profiles of a person. However, it is based on the data and on the insight that was gleaned from it. Various improvements to the proposed approach may be discussed and introduced, new models trained, but due to the nature of the problem, the final step of the study would always be the process of giving meaning to the findings. Otherwise, they are significantly less insightful and way more difficult in use.

In conclusion, this chapter has presented the data, stated the problem, described the theoretical basis for its solution and presented the results of its application in the scope of Personality Identification project. A set of evaluation metrics was defined, according to which various approaches to the solution was judged and compared. Along with argumentation for the final choice of the approach, ways of its possible development and improvement were suggested. Code snippets for used algorithms may be found in Annex A. Final part of the project consists in packaging the "insight producing" code into a usable and simple interface, which is described in section 4.3 on containerization of application.

3. Chatbot development

3.1 The main idea and description of bot's desired capabilities

In this chapter the work conducted in the scope of chatbot project will be described. There is no proper, solid theoretical basis for chatbot development, however, few fundamental concepts are necessary to know for anyone, who is going to start this endeavor, and so they will be presented. A brief historical background of the concept of intelligent artificial agent will be given. Dialog structure will be explained. In the final section of the chapter the process producing a search result based on user's input parameters is described.

To begin with, chatbots are, essentially, software applications that are used to conduct a dialog with users on-line. A dialog may be presented in the form of chat, or a conversation incorporating text-to-speech or speech-to-text technologies. By performing its duties, chatbot replaces a human agent who would typically help clients with their issues.

Capabilities of a bot may vary significantly: form the most simple systems that are only able to detect keywords and respond with predefined phrases to more sophisticated ones that involve extensive natural language processing and understanding algorithms.

The idea of a mechanized agent, that would impersonate a human and perform a set of tasks, was first contemplated by Alan Turing. It was studied and developed in his work "Computing Machinery and Intelligence" [27]. The paper starts with a question that is still bothering greatest mathematicians, computer scientists and philosophers of today:

"I propose to consider the question, 'Can machines think?'"

Following the question, Turing postulates a test, which is also called the "Imitation game" or eponymous Turing's test. This test actually changes the above question from "Can machines think?" to "Can a machine do what thinkers (humans) do ?".

There are nine common objections against "thinking" artificial intelligence that were stated both before Turing's paper and after, as a reaction to it. Among those, one may find Penrose's mathematical objection, Lady Lovelace's statement that computers are incapable of originality or even religious critique of the concept.

Nonetheless, the test has become the industry standard for evaluating AI's capabilities to think and act similar to human agent. Multiple studies, designed to test the abilities and behavior of modern chatbots, such as [28] were conducted recently. However, the general conclusion is that chatbots with their current performance are not quite there, as humans succeed in distinguishing human agent from a chatbot in the majority of cases.

Despite the fact that, at the moment, a bot can not replace human entirely, there is still much to gain from its introduction, especially in a well-defined business cases, such as customer support or order placement.

Current endeavor is one of those cases. As it was discussed in chapter 1 initial design of the chatbot was different from the one that was agreed upon in the end. Thus, only the final design and set of capabilities of the bot will be discussed in this chapter. It is important to mention, that although it was called final, chatbot development is a continuous process, which includes multiple phases of development, testing in the real-world environment and analysis of the performance.

First of all, chatbot was developed with an objective to be deployed on one of the group's site — https://www.trianon-residences.fr/. Its main functionalities are:

- Answer user's questions regarding the enterprise, its underlying principles;
- Answer user's questions on the available programs, special offers;
- Facilitate user's search of a desired apartment, conducting a survey following the defined criteria;
- Based on the survey results, produce a link that presents the apartments from the database that satisfy the criteria;
- Posterior analysis of the performance.

First two abilities aim two make navigation around the site easier for the user, while two latter aim to present a unique functional to present a customized search results.

It is important to understand that the principal goal of a chatbot, both this particular chatbot and in general, is to enrich user experience and to offer an additional entry point to one or another service. So, while user can look for what they are looking for by exploring the site and browsing the menu, another option is available — to ask for bot's help.

While there is an emerging body of research concerning user adoption and use of chatbots, there is a lack of theoretically grounded studies detailing what constitutes good or poor chatbot user experiences [29]. In this paper, authors present findings from a questionnaire study involving more than 200 chatbot users who reported on episodes of chatbot use that they found particularly satisfactory or frustrating.

To sum up, a chatbot, while may not currently represent an intelligent agent, is, potentially, a very important asset in matters of business. It may improve user experience, expand user database, being more convenient in use for those, who would have left the page or application otherwise, or replace human agent in a well defined environment. Whilst commencement of the research in the field may be dated back to 1950s, it is still very vivid in the late years. Along with development of technologies and increase of computational powers, many new possibilities are becoming available for the augmentation of chatbots' abilities, and, thus, for their approach to human's abilities, at least in some particular cases.

3.2 Dialog structure, entities and intentions

This section will present the fundamental concepts of chatbot development and elaborate them for the studied case. Those concepts are:

- Dialog — definition of a set of rules, used by the bot to produce a response, based on what it believes the user wants. The dialog flow is most commonly represented graphically as a tree.
- Intent — an anticipated reason for user's interaction with bot; a goal that user aims to achieve when engaging in a conversation with bot. It is the key component for the development of the conversation. Let one imagine a dialog tree, an identification of an intent will then take place on the lower depth of the tree, closer to the root.

- Entity — a term or an object that provides context for an intent. It may as well be treated as a variable, which bot aims to detect and store its value during conversation, and which users should provide to fulfill their intent.
- Skill — a container for all of the above plus necessary artificial intelligence algorithms that enable bot to process natural language and to work with it.

On practice, one treats the above points as a set of components that should be furnished in a proper way for the development of a bot. It is most prudent to start with dialog, as it is a backbone of the program.

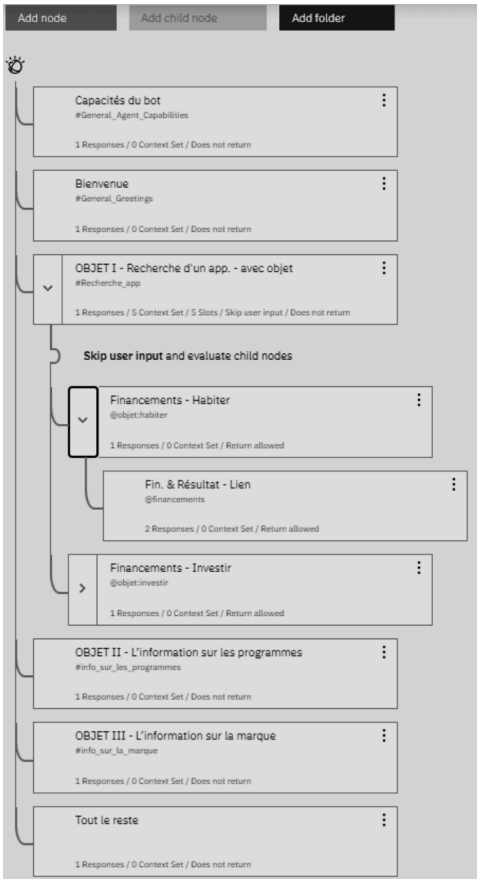

Figure 3.1 Dialog structure — IBM Cloud

The above figure depicts the dialog structure of a developed bot as it is presented in IBM Cloud interface. However, in most cases it is more convenient to analyze the dialog in a conventional tree form (Fig.3.2). One may a notice, that nodes "Bot capabilities" and "Everything else" are not present in Fig. 3.2. The reason for that, is those are not user's intentions, but rather a kind of "Help" nodes. They are

defined with a goal to describe the possibilities of the bot and to help user modify his statements in case bot does not recognize anything familiar. They may be triggered in any moment of the conversation.

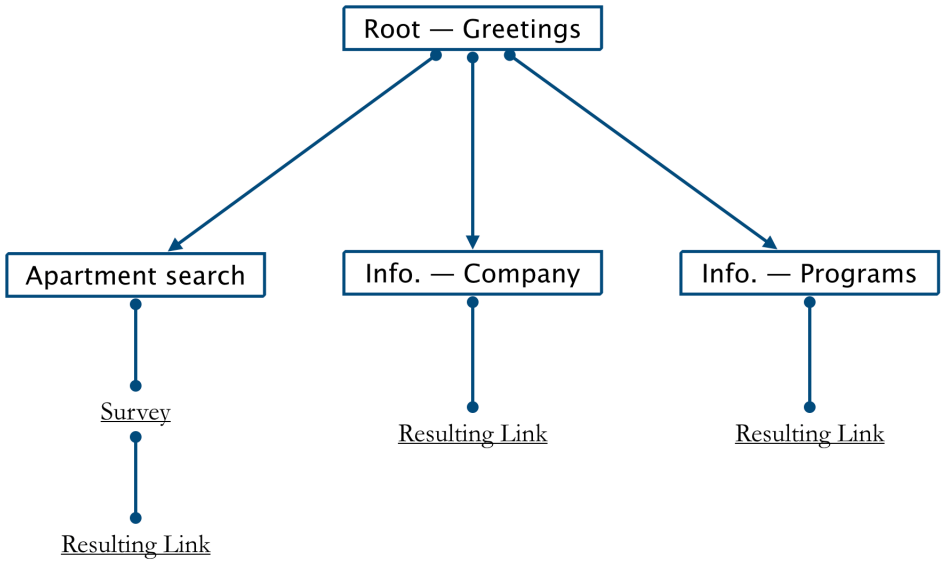

Figure 3.2 Dialog structure — tree form

Due to the bot was meant to be deployed on the company's site, leaves of all the branches of conversation always propose a link, which leads user to the page that was searched for.

From the 2 figures above, one derives three intents that were defined for the current purposes of the bot: "info_sur_la_marque", "info_sur_les_programmes" and "Recherche_app". One may as well observe them in the assistant interface in Fig.3.3.

Intents (13) ↑	Description	Modified ↑↓	Examples ↑↓
#General_Negative_Feedback	Exprimer des commentaires défavo...	a month ago	20
#General_Positive_Feedback	Exprimer un sentiment positif ou d...	a month ago	20
#General_Security_Assurance	Exprimer ses inquiétudes concerna...	a month ago	0
#info_sur_la_marque		a month ago	1
#info_sur_les_programmes		a month ago	2
#Recherche_app		a month ago	11

Figure 3.3 List intents

Besides the defined ones, a set of standard intents was imported to enable bot with additional features, e.g. recognize positive/negative feedback, describe its capacities, etc. Each intent should be supplied with sufficient number of examples of user input, that should trigger its recognition. Figure 3.4 shows a few of these examples for the apartment search intent.

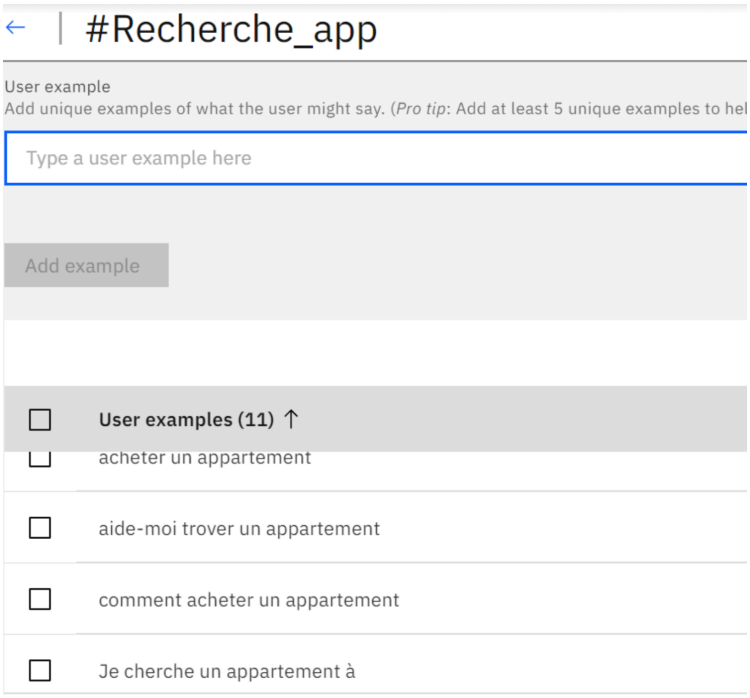

Figure 3.4 User input examples for apartment search

Some of these examples were taken from Google Trends, as the most searched phrases, including "buy an apartment", while others are slight modifications or permutations of the latter.

After bot understood that user is looking for an apartment, he will try to collect all the information that he is able to. This step is marked as "Survey" in Fig. 3.2. This survey is exactly when the entities come into play. Bot asks user a set of questions, such as "Do you have any preference regarding the exterior space ? (parking, garden, balcony…)" or "Are you looking for an apartment to live or to invest in ?" with an aim to determine the search criteria.

After it is done, each entity will contain a value, recognized by the bot as belonging to this entity. This value is stored and may then be accessed to perform any necessary processing. A set of defined entities is given in Fig. 3.5:

Entity (7) ↑	Values	Modified ↑↓
@Espace_ext	Parc, Parking, Jardin, Balcon, aucun, Terrasse	9 days ago
@financements	TVA 5,5, PTZ, aucun, Pinel, pas de pinel, PLS	9 days ago
@finition	clé en mains, PàD	a month ago
@localisation	bas-rhin, alsace, Haut-rhin	9 days ago
@niveaux	Attique, En étage, aucun, RDC, RDJ	9 days ago
@objet	investir, habiter	a month ago
@typologie	1piece, 5pieces, 4pieces, 3pieces, 2pieces	9 days ago

Figure 3.5 Entities list

Same as with intents, a number of synonyms should be provided for each entity, so the bot has sufficient vocabulary for the specific domain that he will work in. An example of that is shown in Fig. 3.6 for the entity "typologie" that corresponds to the configuration of the apartment:

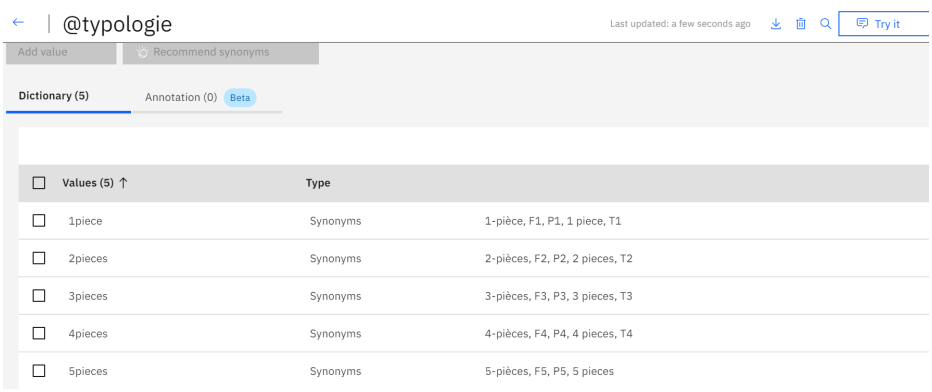

Figure 3.6 Entity "typologie"

It is very important to select possible values carefully, in such a way, that all the necessary cases are covered but also avoiding redundancy, which may cause confusion between different values of the entities. Thus, logically, one should avoid such questions, where a possible response from the user might be just "No". Such response will most likely be treated wrongly. To help with this issue, Watson allows to enable disambiguation, which is an option that allows correction of entities or intents,

recognized by the bot, during training phase, before bot is deployed. An example of that is shown below:

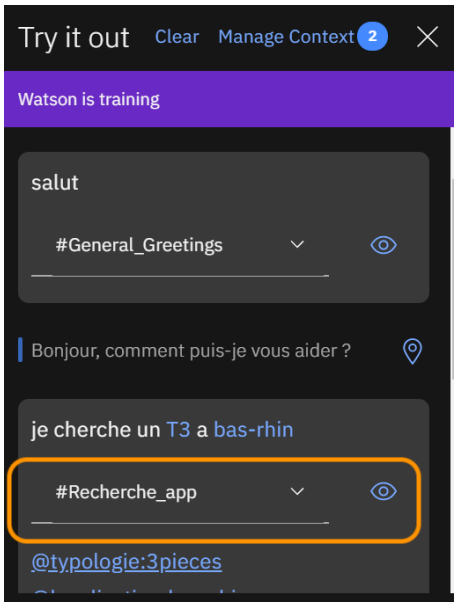

Figure 3.7 Chatbot training

In this example, bot did not recognize the intent correctly. Probably, because of the multiple entities present in the phrase, it gave them the preference. Such behavior is normal during the development of the bot. In such case, one should manually direct a bot to recognize, or rather, prioritize the correct intent in the drop-down menu.

Finally, according to the proposed definition of "Skill", all of the above is packed neatly into a container that is called so (Fig 3.8):

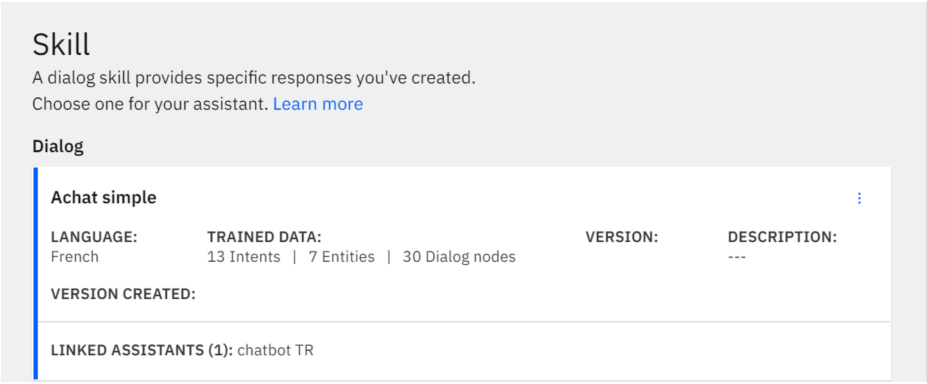

Figure 3.8 Chatbot's skill

Skill may be connected to multiple assistants, enabling same abilities for them. It is stored as a JSON file and may as well be imported or exported.

Development of a bot is a continuous process. A significant chunk of the work will be discovered only when the first version is deployed, due to only then it will encounter real-world users, not those who designed it, and so, previously unknown flaws will be unveiled.

A final version of the chatbot was described in this section. Every essential concept of bot development is covered and described on the studied instance.

Although a new, slightly modified design for the next version was presented during the internship, it was not discussed and thought through, so it may not be considered stable and completed. A draft of the conversation flow for this version may be found in Annex B. Main changes of the next version are branching of the "Apartment search" intent into 4 possible developments, depending on how the user stated the desire to find an apartment. It also includes new entity "Budget", which, unsurprisingly, contains the range of prices that client is willing to pay for an apartment.

Next development step of this project is creation of an application that enables bot to query the database and produce search result for the user. This process and the following deployment on site will be described in the next section.

3.3 Node Red query constructor and deployment

As soon as dialog design is completed, entities and intents are defined and the bot is able to conduct a conversation, one may proceed to setting up the process of exchange between bot and the database.

This phase of the development was the most lengthy and even baffling for a number of reasons. First of them was selection of the database, which should have been able to supply the bot with necessary information. Three options were considered:

1. Database on the target site (https://www.trianon-residences.fr/);
2. Database of an aggregator site (https://www.my-invest.immo/);

3. Database on the company's CRM server.

Unfortunately, every option from the list presented a number of disadvantages. For instance, while aggregator site has the most extensive database, it is used exclusively by real estate agents and so demands authorization to query its database. That would present a showstopper, which may not be eluded, as any user who would talk to chatbot would also need credentials in their disposal to view the search result. That is, of course, impossible. Besides that, it would also look as a minor discrepancy if user would be redirected from company's to a third party site. Thus, this possibility was disregarded first.

CRM database has all the apartments of the group and the information on them is most descriptive, but it is located on a separate server, which inevitably makes querying a bit longer. Moreover, this is an SQL database, with fields containing numeric or text information; it does not have any images or any other kind of content that is more appealing to a client than a number in a cell.

All in all, it was decided that the best outcome will be achieved if database on the target site is used: it has an engaging way of presenting apartments, including possibility of virtual visits and the resulting page is most coherent with the landing page. It was chosen despite the fact, that it does not have the whole set of available apartments at its disposal, but only those that were added to the site. Furthermore, not all the parameters, e.g. exterior space or level, are stored in this case.

Both these issues were addressed during conceptualization and discussion on the approach and it was decided that the site should be augmented with new capacities — a more detailed search, including parameters ($\Longleftrightarrow$ entities in terms of chatbot) and full copy of the apartments database. However, due to the uncertainty of the times, these modifications did not move further than the planning.

Nonetheless, a full cycle of the chatbot development was successfully completed: a stable version of the bot is deployed on the site and it is able to exchange information with resources at its disposal to produce a usable result.

Development version of the site was used for testing purposes and following deployment. An example of conversation, containing bot's response with url, based on user's criteria, may be examined in Fig. 3.9.

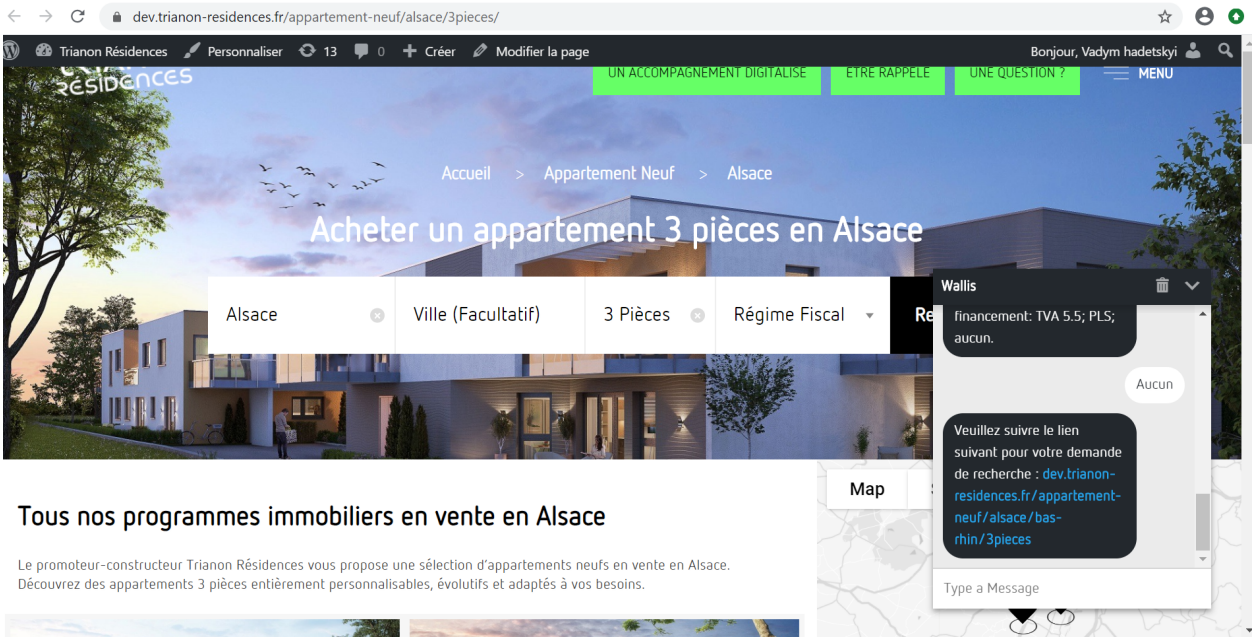

Figure 3.9 Chatbot "Wallis", deployed on dev. version of company's site

To produce resulting link a Node Red application on IBM Cloud was created (Fig. 3.10). Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things [30].

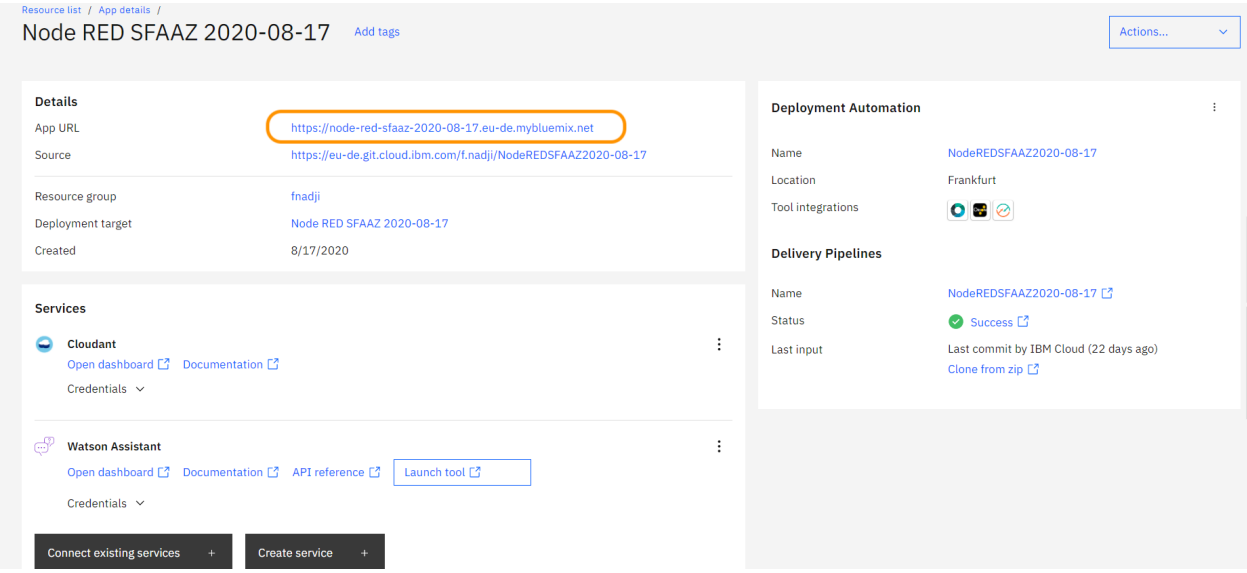

Figure 3.10 Node Red application on IBM Cloud

The application's url is marked with orange on the above picture. It may be accessed via this, or child addresses. A chatbot is connected via a web hook to the

address "https://node-red-sfaaz-2020-08-17.eu-de.mybluemix.net/watson", which is a POST http endpoint set up in Node Red. When "Survey" is completed, chatbot forms a message in the form of JSON, with following variables:

- Typo ∈ {1piece, 2pieces, 3pieces, 4pieces, 5pieces};
- Finance ∈ {TVA 5.5, PTZ, PLS, Pinel, Pas de pinel, Aucun};
- Localisation ∈ {Alsace, Bas-Rhin, Haut-Rhin}.

As it was mentioned above, Node Red is a flow-based application, so the created flow may be observed in Fig. 3.11:

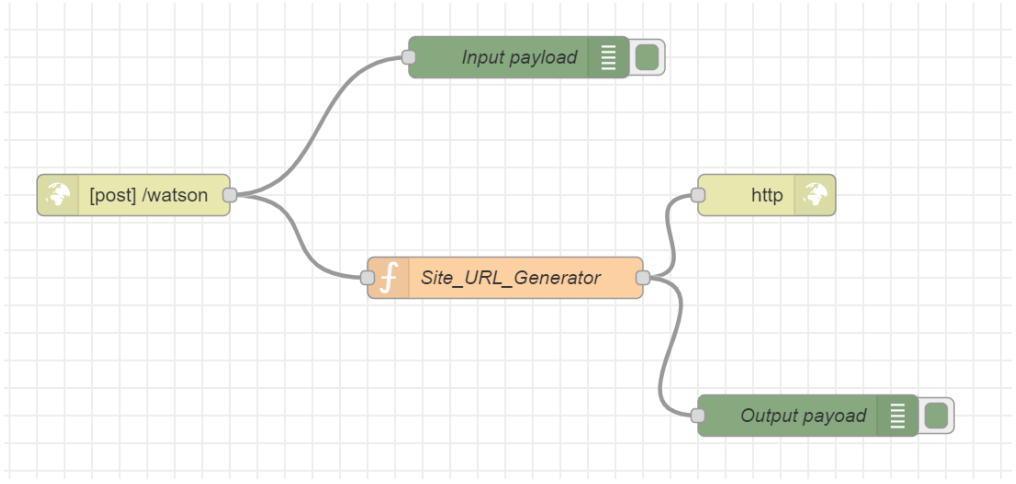

Figure 3.11 Node Red flow

It should be interpreted as follows: http in node accepts POST requests from the bot, sends message further for the processing in the "Site_URL_Generator", which is a javascript function, unpacking the message and constructing a response that is sent to the response http node that answers to the bot. Green nodes are debug ones — they allow one to examine the current content of the message.

An example of the exchange and the http node menu are shown in Fig. 3.12. The code that performs processing of the incoming message and generation of the response url inside of the "Site_URL_Generator" node, may be found in Annex B.

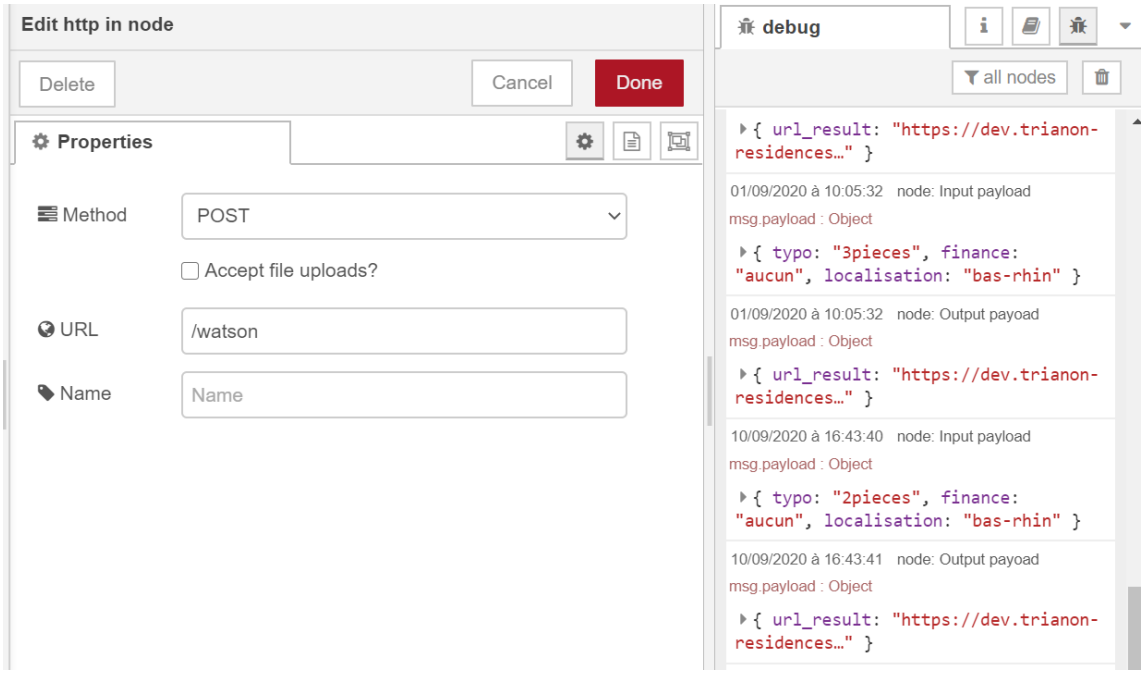

Figure 3.12 Node Red information exchange with Watson Assistant

One should remark the efficiency of the communication, which may as well be observed in Fig. 3.12 as timestamps in debug window. Due to application only works with string types, its operation time is almost negligible. Node Red is an essential component in chatbot development as it was conceived exactly for such purposes. It is recommended by IBM as an extension for Watson Assistant in cases where custom processing of the information is needed.

To sum up, one must say that despite numerous obstacles a full iteration of chatbot development was completed, a working prototype is deployed on the site. It should now be tested, analyzed and modified, because of the process of development is a continuous endeavor. The ensemble of conducted work may also be regarded as a pipeline, a sequence of steps that should be completed for any further version of the bot. A follow-up for this project streams directly from the issues, encountered in the development. Thus, it may be stated with confidence, that to augment the abilities beyond those of the current version of assistant, a number of improvements should first be introduced to the site. Another possibility for the continuation of the project may be creation of the "knowledge base", which would contain domain-specific information that bot is able to utilize to further enlarge its capabilities, moving closely to the performance of a human agent.

4. Cloud Infrastructure

4.1 Cloud computing

This chapter is intended to present the development of the aforementioned projects rather from a technology, back end or infrastructure point of view. Both personality identification and chatbot projects were designed around the functionality of IBM Cloud platform.

Thus, it is quite logical to give it deserved credit and also to shed some light on the concept of the cloud computing and the process of development for a cloud platform.

IBM Cloud was originally chosen by the enterprise for its wide range of services, such as Watson Assistant or Personality Insight, able to boost business or research possibilities. A brief but conclusive study and analysis of available alternatives has shown that there is, indeed, no better choices for the current needs of the enterprise.

The platform is represented by a set of cloud computing services for business, offered by the information technology company IBM. Similar to Google Cloud Platform or Amazon Web services, it combines platform as a service (PaaS) with infrastructure as a service (IaaS). The platform scales and supports both small development teams and organizations, and large enterprise businesses. It is globally deployed across data centers around the world [31].

IBM's first steps towards cloud computing may be dated back to 1960s, when a virtual machine was developed on CP-40 and CP-67 operating systems. Nonetheless, 2014 may be considered a year when IBM Cloud, completed with an acquisition of Cloudant, became available in a form that is close to its current state.

An overall, more defined trend towards cloud computing started to appear in the beginning of twenty-first century. First, this idea manifested itself on a significantly smaller scale: in the form of colocations. Colocation gave users the financial efficiency of renting physical space, while investing in data center real estate would not be available in many cases.

However, with continuous technological advancement the availability of high-capacity networks, low-cost computers and storage devices as well as the widespread

adoption of hardware virtualization, service-oriented architecture and autonomic and utility computing has led to growth in cloud computing [32]:

- In 2006, Amazon created Amazon Web Services;
- In 2007, IBM announced a partnership with Google to promote cloud computing in universities
- In 2008, Google released the beta version of Google App Engine;
- In early 2008, NASA's OpenNebula, enhanced in the RESERVOIR European Commission-funded project, became the first open-source software for deploying private and hybrid clouds, and for the federation of clouds [33];
- In 2008, the U.S. National Science Foundation launched the Cluster Exploratory program to fund academic research using Google-IBM cluster technology in the matters of processing of massive amounts of data;
- In 2010, Microsoft released Microsoft Azure;
- In 2011, IBM

One may conclude, that the interest towards this technology, or, rather development approach, is most vivid. Cloud computing, as addressed by such technological giants mentioned above, and, consequently, designing and development for cloud are based on five concepts (Fig. 4.1).

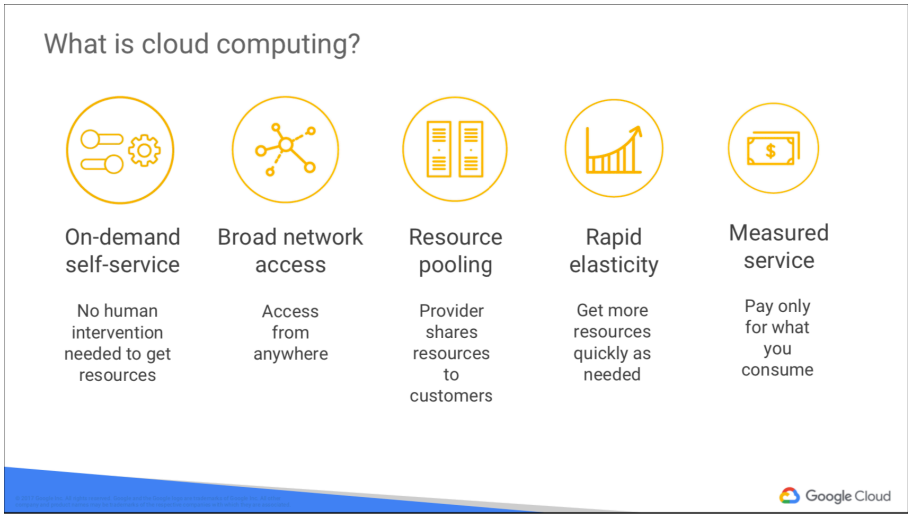

Figure 4.1 Cloud computing fundamental attributes — [32]

Virtualized data centers of today share similarities with the private data centers and colocation facilities of past decades. The components of virtualized data centers match the physical building blocks of hosted computing—servers, CPUs, disks, load balancers, and so on — only they moved to virtual devices. With virtualization maintenance of the infrastructure is still necessary — it is still a user-controlled/user-configured environment. However one may perceive a need for maintenance, this environment provides a number of benefits, such as:

- Speed of the development;
- Low level security;
- Flexibility;
- Portability;
- Scalability.

To sum up, cloud computing exhibits a deep and wide pool of possibilities to enhance the technology capabilities and the efficiency of its very development. It has been of major interest both to tech giants and small enterprises due to its scalability and economic gains. Hence, it was decided that it would only benefit the projects to conduct their development aspiring to incorporate these technologies and to make them at least cloud-compatible if not native to cloud.

4.2 IBM Cloud and services used in the scope of the internship

This section presents services and possibilities of IBM Cloud that were explored and studied during the internship. At the highest level, one may define three main branches of cloud infrastructure (Fig. 4.2). Each of them contains multiple instances and may be hierarchically divided further, but that is outside of the scope.

As it was mentioned in the previous section, IBM Cloud combines PaaS and IaaS. First one (platform as a service) almost exclusively corresponds to “Services” part, while the second one (infrastructure) is a combination of “Storage” and “Applications”.

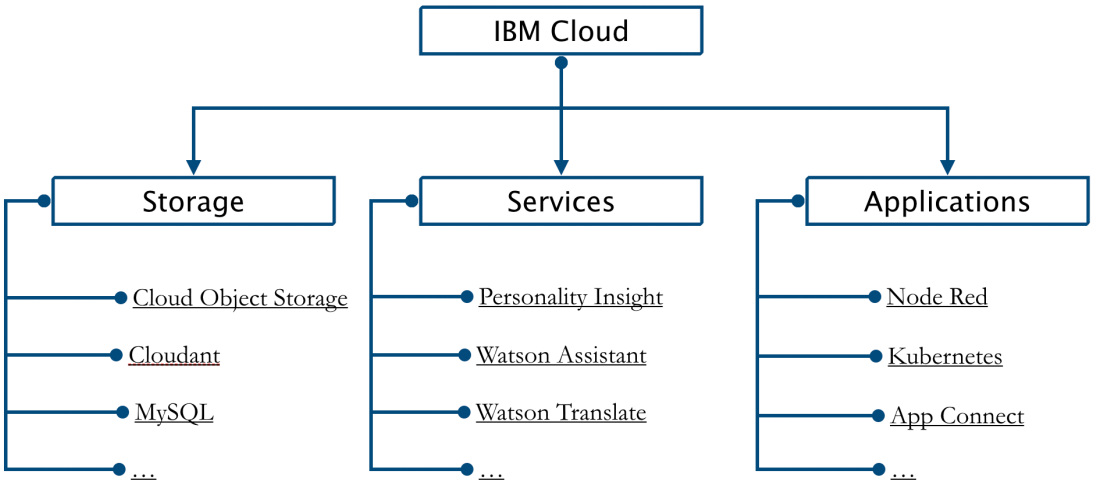

Figure 4.2 IBM Cloud Infrastructure

Storage is a fundamental issue to be covered when developing any project as it is, literally the definition of way the information is stored and kept safe. IBM Cloud offers multiple options, fitting for management of SQL, NoSQL or object databases. Due to the fact that main databases of the enterprise (those that are needed for chatbot queries) are stored outside of the cloud (on CRM) and were not meant to be transported, creation of SQL or NoSQL database was not necessary. On the other side, in the frame of personality identification project, one had to work with various kinds of data: text, data frames, JSON files, python scripts and notebooks. For such case, one should use object storage, which, complying with its own name, allows to store objects of various kinds. It is similar to the way files are stored on usual OS, though, there are differences, mainly in access and grouping of files. It is done thanks to the introduction of "buckets", which are groupings of files, which may be accessed by their key and used by other services and applications.

So, Cloud Object storage was created for all the purposes of storage in the scope of the internship.

Services is, certainly, the most intriguing thing that any cloud offers. They should be regarded as possible accelerators and ways to improve the project in development.

Personality Insight was the first one studied. It uses linguistic analytics to infer individuals' personality characteristics, including Big Five, Needs, and Values, from digital communications such as email, blogs, tweets, and forum posts. The service can

automatically infer, from potentially noisy social media, portraits of individuals that reflect their personality characteristics. The service can infer consumption preferences based on the results of its analysis and, for JSON content that is timestamped, can report temporal behavior [31].

Several researchers found that variations in word usage in writings such as blogs, essays, and tweets can predict aspects of personality. IBM found that people with specific personality characteristics responded and retweeted in higher numbers in information-collection and -spreading tasks. For example, people who score high on excitement-seeking are more likely to respond, while people who score high on cautiousness are less likely to do so. Similarly, people who score high on modesty, openness, and friendliness are more likely to spread information [34].

From a technical point of view, the service first tokenizes the input text to develop a representation in an *n*-dimensional space. An open-source word-embedding technique, GloVe, is used to obtain a vector representation for the words in the input text. Then this representation is fed to a machine-learning algorithm that infers a personality profile with Big Five, Needs, and Values characteristics. To train the algorithm, scores from surveys that were conducted among thousands of users along with data from their Twitter feeds were used.

As for the accuracy of model's prediction, IBM has conducted a validation study and gathered few thousands of user's responses to a survey to establish ground truth. It is stated that for English language average MAE is approximately 0.11 and average correlation between inferred and actual scores is around 0.3. Thus, one should take that into account and be prudent when making conclusions based on the gained insight.

In the scope of the same project, a lot of development was done in Watson Studio. This service functional is fairly simple — it should be connected to a storage instance, then it enables one to operate in an environment, similar to usual OS: one may work with files, create code snippets in the form of scripts or notebooks and also apply specific processing pipelines, suggested by IBM (see Annex C).

Finally, it was desirable to be able to access trained ML models without having them available locally. It is practical especially for distributed cases, or when an

application is not thought to be running continuously on a single device. To solve this issue, Watson Machine Learning service was examined.

This service was created with an objective to accelerate AI and machine-learning deployment. With its open, extensible model operation, Watson Machine Learning helps businesses simplify and harness AI at scale across any cloud. Watson Machine Learning provides capabilities to [31]:

- Deploy models built with IBM Watson Studio and open source tools.
- Dynamically retrain models
- Automatically generate APIs to build AI-powered applications
- Streamline model management and deployment end-to-end

Although it is presented as a very powerful tool, some difficulties were encountered with one of deployments, which would not be explained by any forum or documentation (Fig. 4.3). Model's status kept repeatedly changing to "UNKNOWN", without any changes to the code or model's exploitation.

A significant part of the internship revolved around the functionality of Watson Assistant. It is a service that combines machine learning, natural language understanding, and an integrated dialog editor to create conversation flows with users.

The capabilities of a bot, developed with this service, are very impressive: one can create special skills, such a s "Search skill" which can be used by a bot to search given source and extract necessary information upon triggering a specified condition. Besides that, assistant can be connected to any external application using web hooks and html requests. That means bot can exchange information with the application, which can process it in any way one needs.

Besides, the ability to develop the chatbot, Watson Assistant also facilitates its deployment. It supports plethora of embedding applications and web technologies, which makes it, indeed, an unexpectedly painless process.

It is worth to mention other services, such as Watson Discovery (NLU), Cognos Dashboards (dashboard creation), Streaming Analytics (dashboard creation), Cloudant (storage), that were studied during various phases of the internship. First three services were potential candidates to solve corresponding tasks, but did not

prove to fit the objective entirely. Cloudant, however, was used collaterally as it was necessary for creation of Node Red application.

All the services may be connected between each other or with external sources, in most cases. As with Watson Assistant, it is mostly done through the exchange of html requests. In other cases, services may be connected via url, api key and credential information. This information from one of the services should be specified for the other, which will try to establish a connection, as depicted in Fig. 4.3 for Watson ML.

Figure 4.3 Watson ML connection

Cloud offers graphical interface to create, maintain and access credential information on storage, service and application instances.

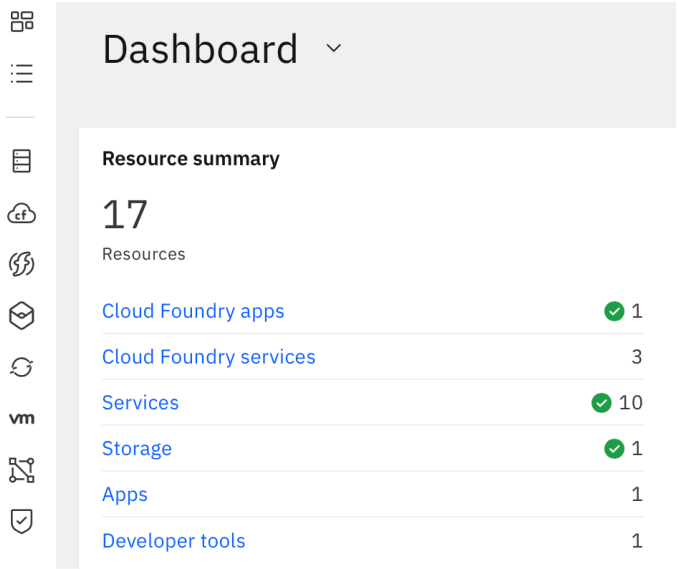

Figure 4.4 IBM Cloud resources access

One should remark that despite the fact that graphical interface is usually very practical and easy to navigate, it may become overcrowded fast enough. Moreover, the user-friendliness of the interface may be subjectively considered doubtful. In such cases, one may explore the possibilities of IBM Cloud CLI, which seemed slightly greater than those of a graphical interface.

```
Vadyms-MacBook-Pro:FlaskAndDeploy thewaveorthemountain$ source app_env/bin/activate
(app_env) Vadyms-MacBook-Pro:FlaskAndDeploy thewaveorthemountain$ ibmcloud
NAME:
  ibmcloud - A command line tool to interact with IBM Cloud
  Find more information at: https://ibm.biz/cli-docs

USAGE:
  [environment variables] ibmcloud [global options] command [arguments...] [command options]

VERSION:
  1.1.0+cc908fe-2020-04-29T09:33:25+00:00
```

Figure 4.5 IBM Cloud CLI

Command line interface may be less practical, but it was proven to be irreplaceable, for instance, when working with Kubernetes and cluster settings.

In conclusion, IBM Cloud is a very powerful platform that empowered and accelerated the development process during the internship. It offers tools and services that can not be replaced in the frame of some projects, simply due to the unavailability of data or computing capabilities. Besides that, all these tools are gathered under the same roof, which defines their management, maintenance and connection in a universal, standardized way, making these processes significantly less fastidious and time-consuming.

On the other hand, one must mention, that cloud is also a dynamic, evolving environment and, although universal among platform's subparts, the standards of development or maintenance processes constantly change and evolve, — thus, continuous modifications should be made. This fast pace of the evolution of the platform, unfortunately, finds its reflection in the lagging, incomplete or entirely obsolete documentation, which is, especially, complicated during learning and explorative stages of the work.

4.3 Personality Insight project delivery and containerization

The final section of this chapter aims to present efforts made to furnish the most convenient and usable way to gain insight on person's personality with an approach described in chapter II of the given report. This information is placed in the current chapter due to, as was mentioned earlier, the whole development revolved around the infrastructure of IBM Cloud, and other cloud platforms. Thus, it is logical that it belongs to the corresponding chapter.

First of all, it is important to remark that the idea for the packaging of all the steps to perform in the scope of personality identification into a single application emerged from the practical necessity to convey the intermediate results in a more understandable kind of way. A number of meetings was hold on to discuss the progress, issues and next steps in the development of the projects, during which it became clear that it is not at all straightforward and easy to explain how one applies the proposed personality identification approach and how one should glean any new knowledge from the findings. A simple input-output application and the dashboard-like presentation, containing multiple visualizations to allow different point of view on the subject, of the result was decided to be a good solution. On the other hand, some might classify the development process of such an application as the final step of end-to-end development of a machine learning project.

With the commencement of the work on this part of the project an extensive study of IBM Cloud's catalog and documentation was conducted. Few possible ways to proceed were considered, such as those, involving dashboard creation service Streaming Analytics and Cognos Dashboards. They were found to be too rigid, allowing few customizations. In fact, fairly complicated pipeline of data transformation, that was to be implemented, very much clipped the set of available solutions, dictating the terms for the available ones.

The data pipeline is presented in the Fig. 4.6. As we can see, the result is presented in the form of a dashboard page, which is a web page of a predefined structure that presents the obtained result. It was decided that a simple two-page (input-output) web application was the most reasonable choice, due to the input, namely, text, is easy to copy and paste in the input form, and the prevailing majority

of the office workers have both browsers installed on their machines and skills to operate with them.

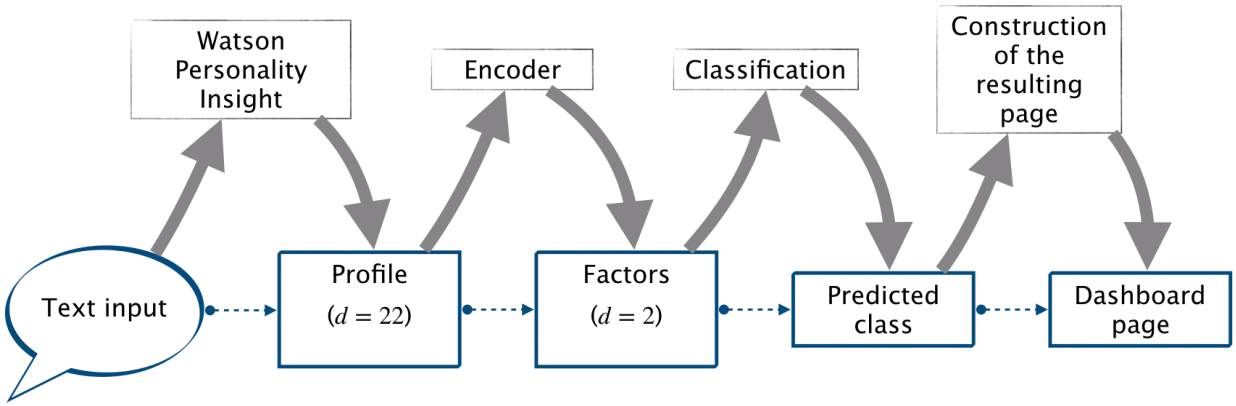

Figure 4.6 Data pipeline

It is clear that any web application needs a server to be launched on. This issues is addressed by incorporating Flask library for python, which manages the server-side and allows to serve html templates. Upon receiving the input, application sends it to a python function that comprises the whole data pipeline, and responds with the result page. However, solving one issue, such approach creates a set of very important ones to resolve:

- All the used python code needs a number of dependencies to compile. How to ensure the running machine will be able to do it ?
- Web application needs a strictly defined arborescence of the launching directory, otherwise, wrong pages may be served.
- Running OS must be allowed and configured to start a server, necessary ports must be available. How to enforce this constraint ?

Fortunately, all this issues are resolved by containerization of the application. It was performed using Docker. Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers [35]. Containers should be viewed as isolated, self-sufficient bundles of software, libraries and configuration files. To instantiate a container one should create a Dockerfile in the directory that is going to be containerized, under the name “Dockerfile” and put all

the commands that should be run upon launch of the virtual machine. An example of this file for the current application is given in Fig. 4.7.

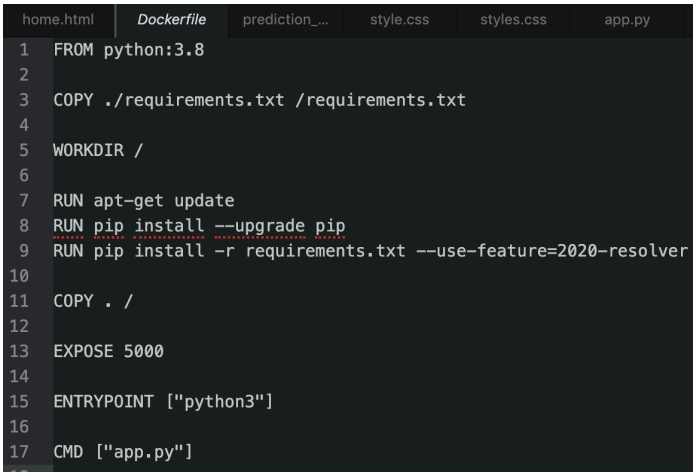

Figure 4.7 Dockerfile

It is very important to construct "requirements.txt" in a very thorough manner. It should inform the machine on the versions of the dependencies that should be installed. Minor discrepancies may cause conflicts between incompatible versions of the libraries and undermine the functioning of the application inside of a container. The requirements file and the directory, containing everything needed for correct containerization and following functioning of the application are shown in Fig. 4.8.

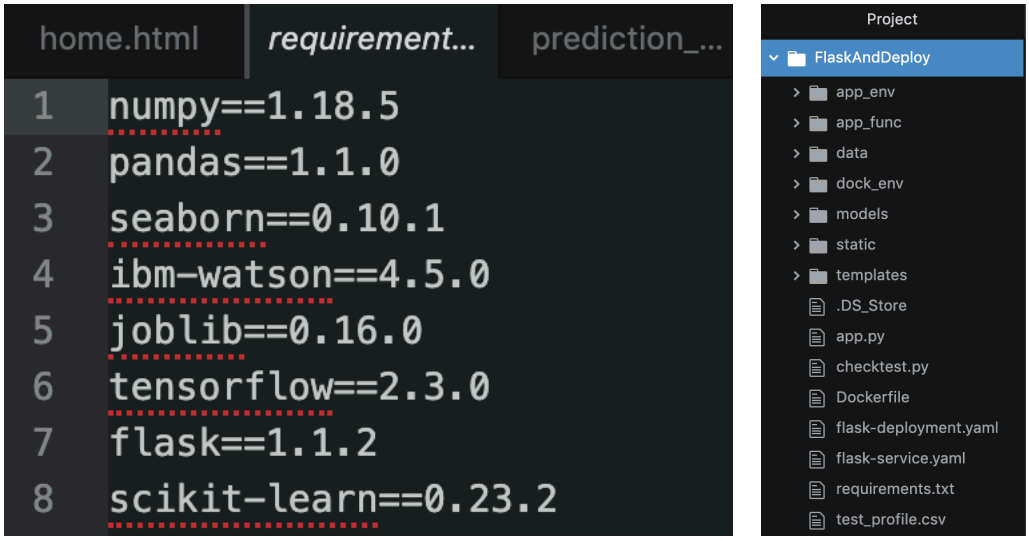

Figure 4.8 Requirements file and application structure

It is noteworthy, that such an approach to development allows the application to be run both locally and on the cloud. After docker container was created, it was deployed on IBM Cloud, using Kubernetes Engine service. This service enables creation and maintenance of a cluster on the cloud. In such a way application can be launched once and left turning endlessly, however, it would not be free. With creation of cluster, one, essentially, borrows a chunk of machinery from the cloud provider, thus, these kinds of services are rarely free. IBM's Kubernetes service was used to create a cluster, though, without subscription cluster exists only during 30 days and then it is automatically deleted. It should be recreated and reconfigured, which is a rather time-consuming process.

The company did not express any desire to move development to the cloud for all the existing projects, so there was no economical profit in paying for Kubernetes services. Besides that, the current application would not have a wide user base — it is supposed to be useful to gain insight on present or potential customers, so marketing, sales and customer support agents are the most likely users. So, application was launched and tested locally both on Mac OS and Windows. The image that docker creates to be an environment for the application is, in fact, Linux, which makes it inherently compatible with all Unix-based OS. All in all, one obtains entirely cross-platform application.

It may be launched by entering the following command into command line both under Mac or Windows OS:

```
docker run –p 5000:5000 personality_app:latest
```

Docker interface may be used as well. Then, the application may be accessed on the localhost (http://0.0.0.0:5000/).

A home page (Fig. 4.9) allows one to enter the text input for the processing. When it is submitted, a result page will be loaded (Fig.4.10). Result page consists 5 elements which may help to gain some information regarding the given profile and a block with links to the articles, which may help interpret the result. Those elements are:

1. Description of the class (cluster) and radar visualization of principal traits (big5 + self-transcendence);
2. Proportions of, or probabilities to belong to, each cluster;

3. Comparison of principal trait plot of profile and all cluster cores;
4. Radar plots of every aspect of the profile;
5. Profile position and positions of centroids in 2D space

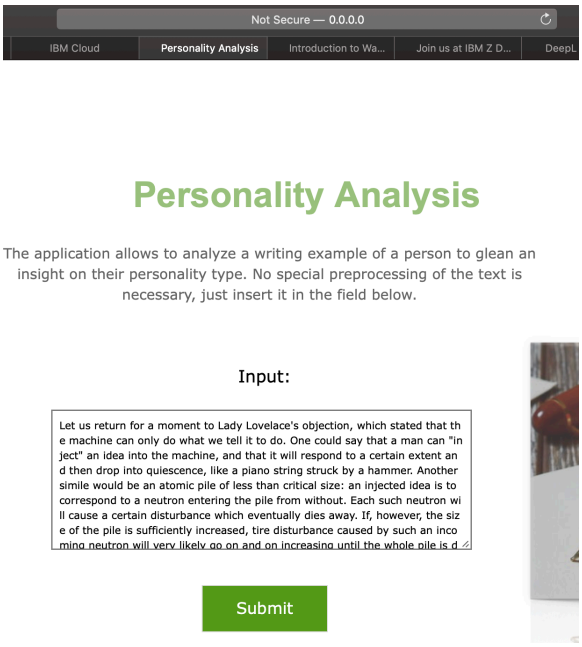

Figure 4.9 Application home page

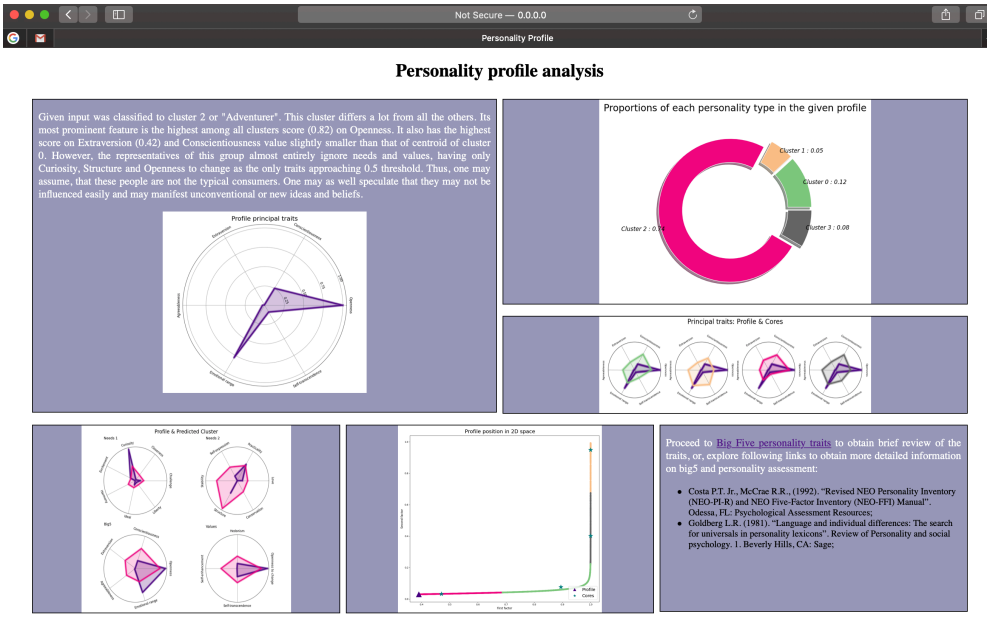

Figure 4.10 Result page

As it was mentioned in chapter II, the proposed approach does not at all pretend to be an exclusively precise and reliable technology that allows to understand how and what humans think and how to make them consume what one wants. Instead, it is intended to be a data-driven tool, augmentation of possibilities to maximize the efficiency of communication regarding a particular matter.

One may remark, that the machine learning models lack data for the optimal performance, the data itself is not sufficiently diverse to cover whole population. The precision of Watson Personality Insight, described in section 2 of the current chapter, should also be taken into account.

Nonetheless, the methodology itself is based on the underlying theoretical studies and supported by a number of extensive practical experiments. The delivery of the results is implemented in a flexible and portable way, such that the application may be easily modified with interface changes, upgraded models or moved entirely to the cloud.

Concluding this chapter, one should say, that developing projects for the cloud presents multiple opportunities to accelerate and lighten the process. However, one should be familiar enough with the infrastructure to profit from those. Significant amounts of work in the scope of both projects of the internship presented a novelty and demanded a lot of studying, trial and error to be implemented properly. The current chapter aimed to present this aspect of the internship as much as possible.

Conclusion

The given report presented the work conducted during the internship. The initial subject and its evolution into two independent projects — "Chatbot" and "Personality Identification" — were described. It was suggested how both of them may be used to enhance the communication between parties, namely, in the domain of user experience and marketing strategy correspondingly.

In the scope of Personality Identification project an exploratory research on the techniques of unsupervised learning was performed, a number of references studied, a set of experiments designed and conducted to compare various ways to approach the problem. A final data pipeline was defined and a methodology was formed, a number of recommendations of how to proceed was given.

As for Chatbot project, it presented a previously unseen space of development of the ideas, with the entirety of the work produced in IBM Cloud. Bot development is not yet a separate discipline, rather it is an endeavor that lays on the intersection of multiple others, such as computer science, mathematics, psychology, etc. It demanded an appropriate state of mind to understand the potential flaws or bottlenecks in the conversation. As in the case of the first project, potential courses of continuation of development and prospects of the project were discussed.

In the frame of both projects, previously existing skills and knowledge in artificial intelligence, mathematics and computer science were put into test and honed. At the same time, a decent stack of technologies, thoroughly described along the report, must have been learnt and applied on practice. IBM Cloud, being one of them, presents a whole new environment to work in and to develop for. It allowed one to form a notion of cloud computing and development for cloud, defining its most noticeable pros and cons.

Finally, both projects reached their essential milestones. Working prototypes were demonstrated, deployed and ready to fulfill the missions of communication enhancement in their corresponding ways. One may speculate, that similar demands will only increase, becoming both more feasible and more powerful with the development of technologies. Early first steps in this direction surely present a sound competitive advantage for the enterprise.

Bibliography

1. Crowne, D. P. (2007). "Personality Theory", Don Mills, ON, Canada: Oxford University Press.
2. Hoijer, H. (1954). "Language in culture: Conference on the interrelations of language and other aspects of culture", Chicago: University of Chicago Press
3. Goldberg L.R. (1981). "Language and individual differences: The search for universals in personality lexicons". Review of Personality and social psychology. 1. Beverly Hills, CA: Sage.
4. Hartman T. (1998). "The color code", New York : Scribner.
5. Myers, Isabel Briggs with Peter B. Myers (1980). "Understanding Personality Type", Mountain View, CA: Davies-Black Publishing.
6. Hinton G., Sejnowski T. (1999). "Unsupervised Learning: Foundations of Neural Computation". MIT Press.
7. Ester M., Kriegel H-P., Sander J., Xu X. (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise." Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press
8. Kriegel H.P, Kröger P., Sander J., Zimer A. (2011). "Density-based clustering". Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1.
9. Steinhaus H. (1957). "Sur la division des corps matériels en parties". Bull. Acad. Polon. Sci.
10. McLachlan G.J. (2000). "Finite Mixture Models". Wiley
11. Dempster A.P., Laird N.M., Rubin, D.B. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". Journal of the Royal Statistical Society, Series B.
12. Aggarwal C.C., Hinneburg A., Keim D.A (2001). "On the Surprising Behavior of Distance Metrics in High Dimensional Space". IBM T. J. Watson Research Center

13. Mahalanobis P. C. (1936). "On the generalised distance in statistics". Proceedings of the National Institute of Sciences of India.

14. Jolliffe I.T. (2002). "Principal Component Analysis". Series: Springer Series in Statistics, 2nd ed., Springer, NY, XXIX

15. Cattell R.B. (1978). "Use of Factor Analysis in Behavioral and Life Sciences". New York: Plenum

16. Cudeck R., O'Dell L.L. (1994). "Applications of standard error estimates in unrestricted factor analysis: Significance tests for factor loadings and correlations". Psychological Bulletin.

17. Goodfellow I., Bengio Y., Courville A. (2016). "Deep Learning". MIT Press

18. Ballard D.H. (1987). "Modular learning in Neural Networks". Department of Computer Science University of Rochester, Rochester, New York

19. Hopkins B.; Skellam J.G. (1954). "A new method for determining the type of distribution of plant individuals". Annals of Botany. Annals Botany Co. 18

20. Davies D.L., Bouldin D.W., (1979). "A Cluster Separation Measure". IEEE Transactions on Pattern Analysis and Machine Intelligence. PAMI-1

21. Rousseeuw P.J., (1987). "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis". Computational and Applied Mathematics. 20

22. Xie J., Girshick R., Farhadi A. (2016). "Unsupervised Deep Embedding for Clustering Analysis". Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA

23. Hopfield J.J., Tank D.W., (1985). "Neural computation of decisions in optimization problems". Biological Cybernetics 55

24. Turing A.M. (1950). "Computing machinery and intelligence". Mind. LIX (238)

25. Costa P.T. Jr., McCrae R.R., (1992). "Revised NEO Personality Inventory (NEO-PI-R) and NEO Five-Factor Inventory (NEO-FFI) Manual". Odessa, FL: Psychological Assessment Resources

26. Schwartz S.H., (2006). “Basic Human Values: Theory, Measurement, and Applications” . Revue Francaise de Sociologie, Vol. 47

27. Turing A., (1950). “Computing Machinery and Intelligence”. Mind, LIX(236)

28. Ishida Y., Chiba R. (2017). “Free Will and Turing Test with Multiple Agents: An Example of Chatbot Design”. International Conference on Knowledge-Based and Intelligent Information and Engineering Systems

29. Følstad A., Brandtzaeg P.B., (2020). "Users' experiences with chatbots: findings from a questionnaire study". Qual User Exp 5, 3, Springer.

30. Node Red — https://nodered.org

31. IBM Cloud — https://www.ibm.com/cloud

32. Google Cloud Fundamentals: Core Infrastructure — MOOC. https://www.coursera.org/learn/gcp-fundamentals

33. Rochwerger B., Breitgand D., Levy E., Galis A., Nagin K., Llorente I. M., Montero R., Wolfsthal Y., Elmroth E., Caceres J., Ben-Yehuda M., Emmerich W., Galan F. (2009). “The Reservoir model and architecture for open federated cloud computing”. IBM Journal of Research and Development, 53

34. Gill A. J., Nowson S., Oberlander J. (2009). “What Are They Blogging About? Personality, Topic and Motivation in Blogs”. AAAI Publications, Third International AAAI Conference on Weblogs and Social Media

35. Docker — https://www.docker.com

Annex A

	[Colours].ETL_Twitter	Python 3.6	ferial nadji	Jun 02, 2020	
	[Colours].PersonalityInsightGen	Python 3.6	ferial nadji	Jun 03, 2020	
	[Colours].Clustering	Python 3.6	Vadym Hadetskyi	Aug 25, 2020	
	[Colours].Clustering_subsets	Python 3.6	Vadym Hadetskyi	Jun 12, 2020	
	[Colours].FA_n_Clustering	Python 3.6	Vadym Hadetskyi	Sep 02, 2020	
	[Colours].AutoEncoder_n_Clustering	Python 3.6	Vadym Hadetskyi	Sep 04, 2020	
	[Colours].Clustering_NN	Python 3.6	ferial nadji	Jul 06, 2020	

Figure 1. Essential notebooks used in the scope of Personality Identification project

```
def gather_user_tweets(query,
                       activity_limit = 10,
                       char_significance=2000,
                       tweets_per_user = 200,
                       pages_to_look=10,
                       REQUESTS_MADE=0,
                       MAX_REQUEST_LIMIT=1480,
                       SLEEP_TIMER=600,
                       verbose=False):

    # Firstly, all available tweets on the given topic (query)
    # are gathered throughout all the pages of search
    # those which fit the constraints are stored in <timelines>
    timelines = {}
    waited = False

    for page in range(pages_to_look):
        try:
            some_users = api.search_users(q=query, count=20, page=page)
        except Exception as E:
            if verbose:
                print("Page {:d} does not exist".format(page))
            break

        if verbose:
            print("Found {:d} users from page {:d}. ".format(len(some_users), page))

        k = 0
        for u in some_users:
            if u.friends_count > activity_limit or u.followers_count > activity_limit:
                k += 1

                # checking and waiting if limit reached
                if REQUESTS_MADE >= MAX_REQUEST_LIMIT:
                    print('REQUEST LIMIT HAS BEEN REACHED. THE EXTRACTION WILL BE PUT ON
HOLD FOR {:.2F} MINUTES.'.format(SLEEP_TIMER/60))
                    time.sleep(SLEEP_TIMER)
                    waited=True
                    REQUESTS_MADE= 0

                # loading this user's timeline
                if not u.protected:
                    timelines[u.id] = api.user_timeline(id = u.id, count = tweets_per_use
r)
                    REQUESTS_MADE += 1
        if verbose:
            print("Gathered {:d} users from page {:d}. ".format(k, page))


    print("<<<>>>")
    print("Gathered {:d} users and their timelines. Moving on.".format(len(timelines)))

    # one needs only text, so
    # for each user the tweets are concatenated
    # and put into a dictionary
    # currently only english tweets are accepted

    user_tweets_dict = {}

    for uid in timelines:

        this_user_tweets = ""
        for t in timelines[uid]:
            if t.lang == 'en':
                this_user_tweets += t.text
                this_user_tweets += '\n\n'

        if len(this_user_tweets) >= char_significance:
            user_tweets_dict[uid] = this_user_tweets

    total_users = len(user_tweets_dict)
    print("Tweets of {:d} users satisfy the given constraints. They are succesfully proce
ssed and saved.".format(total_users))
    if verbose:
        av_corp = sum([len(tweet) for tweet in user_tweets_dict.values()])
        av_corp /= total_users
        print("Average user's corpus length is {:.2f}.".format(av_corp))
    print("<<<>>>")

    return user_tweets_dict, REQUESTS_MADE, waited
```

Figure 2. Twitter extraction function

```
gathered_data = {}
current_reqs_made = 0
try:
    for topic in various_topics:
        print("Searchig for users on '{:s}'".format(topic))
        result, reqs, waited = gather_user_tweets(topic,
                                    pages_to_look=10,
                                    tweets_per_user=500,
                                    char_significance=1500,
                                    REQUESTS_MADE=current_reqs_made,
                                    MAX_REQUEST_LIMIT=1480,
                                    verbose=False)
        if waited:
            current_reqs_made = 0
            print("Requests counter is nullified.")
        else:
            current_reqs_made += reqs

        gathered_data.update(result)
        print("Total users gathered: ", len(gathered_data))
        print("REQUESTS_MADE: ", current_reqs_made)
        print()
except Exception:
    print('Interrupted.')
```

Loading data on disk

It is only left to store the gathered data somewhere

```
# @hidden_cell
# The project token is an authorization token that is used to access project resources li
ke data sources, connections, and used by platform APIs.
from project_lib import Project
project = Project(project_id='222a59b5-e3fe-4929-8ef6-435605997638', project_access_token
='p-5b537f34172609279216be5f8e628426f23fb49d')
pc = project.project_context
```

```
file_name = "twitter_main.csv"
df = pd.DataFrame.from_dict(clean_data, orient='index', columns=['Text'])
df.reset_index(inplace=True)
df.rename(columns={"index": "UID"}, inplace=True)
df.head()
```

	UID	Text
0	89168924	Radio stations and TV channels have changed th...
1	74580436	On Tuesday, June 2nd, Apple Music will observe...
2	52536879	Little Richard receives the Merit Award at the...
3	17243213	In response to George Floyd's death—the death...
4	278662460	3 years ago today, BTS_twt accepted their firs...

```
project.save_data(file_name, df.to_csv(index=False))
```

```
{'file_name': 'twitter_main.csv',
 'message': 'File saved to project storage.',
 'bucket_name': 'personalityinsightproject-donotdelete-pr-1pbhytmlhswznj',
 'asset_id': '522b1343-6e17-4c4b-8f83-510576779af8'}
```

Figure 3. Loop over topics set and saving of the data

Creating dataframe of profiles

```
def get_profile_element(profile, element, output_name=None, plot=True, return_df=True):

    av = ['personality', 'needs', 'values']
    #assert element in av, "can't visualize this element"
    value_field = 'percentile'
    if not output_name:
        output_name = element

    ex_dict = {el['name']:el[value_field] for el in profile[element]}

    df = pd.DataFrame.from_dict(ex_dict, orient='index')
    df.reset_index(inplace=True)
    df.rename(columns={"index": output_name, 0: "scores"}, inplace=True)

    if plot:
        plt.figure(figsize=(15, 7))
        sns.barplot(y='scores', x=output_name, data=df);
        plt.show()

    if return_df: return df
```

```
def profile_elements_keys(text):

    profile = service.profile(
    text,
    'application/json',
    raw_scores=True,
    consumption_preferences=False).get_result()
    print("hey  ",profile['warnings'])

    av = ['personality', 'needs', 'values']
    outs = {}
    for element in av:
        ex_dict = {el['name']:el['percentile'] for el in profile[element]}
        outs[element] = ex_dict.keys()

    return outs
```

```
def generate_profile_df(profile, element):

    ex_dict = {el['name']:el['percentile'] for el in profile[element]}

    return pd.DataFrame.from_records(ex_dict, index=[0], columns=ex_dict.keys())
```

```
def text2profile(text, profile_elements = ['personality', 'needs', 'values']):
    try:
        profile = service.profile(
            text,
            'application/json',
            raw_scores=True,
            consumption_preferences=False).get_result()

        result = pd.DataFrame()
        for el in profile_elements:
            result = pd.concat([result, generate_profile_df(profile, el)], axis = 1)

        return result.iloc[0]
    except Exception:
        pass
```

```
def create_profiles(df,  profile_elements = ['personality', 'needs', 'values']):

    df_uids = pd.DataFrame(df.UID)
    df_vals = df.apply(lambda row: text2profile(row['Text'], profile_elements), axis=1)
    r = pd.concat([df_uids, df_vals], axis=1)
    return r
```

```
# test
dfa = df.iloc[:17].copy()
profs = create_profiles(dfa)
profs.head()
```

Figure 4. Profile (from text) generation

```
def Davies_Bouldin_index(df, cats, cluster_cores, distance, order=1):

    n = len(cluster_cores)
    sigmas = np.zeros(n)

    # average intra-cluster distances
    for i, cc in enumerate(cluster_cores):
        sigmas[i] = np.mean(df[df[cats] == i].apply(lambda row: distance(row[:-1], cluste
r_cores[i], k=order), axis=1))

    # max intra-inter relations
    dists = [np.max([(sigmas[i] + sigmas[j])/distance(cluster_cores[i], cluster_cores[j],
order) for j in range(n) if i != j]) for i in range(n)]
    return np.mean(dists)
```

Figure 5. DBI computation

```
def silhouette_evaluation(df, plot=True, distance='l2',samples_per_class=50, cmap='hsv'):

    X = df.to_numpy()
    labels = X[:, -1]
    X = X[:, :-1]
    cluster_codes, cluster_lengths = np.unique(labels, return_counts=True)

    sil = silhouette_samples(X, labels, metric=distance)
    sil_score = silhouette_score(X, labels, metric=distance)

    res_df = pd.DataFrame(np.stack([labels, sil], axis=1), columns=['Class', 'Silhouette'
])
    sil_means = res_df.groupby('Class').mean()

    print('Silhouette quality of each cluster:')
    print(sil_means)
    print('Silhouette score of the clustering is: ', sil_score)
    if plot:
        sample_df = pd.DataFrame(columns=['Class', 'Silhouette'])
        ax_labels = []

        for cat in cluster_codes:
            sample_df = pd.concat([sample_df, res_df[res_df['Class'] == cat].sample(sampl
es_per_class)])
            ax_labels.append("Cluster {:d}".format(int(cat)))

        # Create a color palette:
        my_palette = plt.cm.get_cmap(cmap, len(cluster_codes))
        sam = sample_df.to_numpy()
        sam_class = sam[:, 0]
        sam_sil = sam[:, 1]

        fig, ax = plt.subplots(figsize=(15, 20))
        plt.title('Silhouette values', fontsize=20)

        plt.barh(y=np.arange(0, len(sam_sil)), width=sam_sil, edgecolor='k',
                 color=my_palette(sam_class/np.max(sam_class)))

        ax.set_yticks(cluster_codes*samples_per_class + samples_per_class//2)
        ax.set_yticklabels(ax_labels, fontsize=13, rotation='vertical')
        ax.invert_yaxis()  # labels read top-to-bottom
        ax.set_xlabel('Silhouette', fontsize=15)


    return res_df, sil_means
```

Figure 6. Silhouette computation

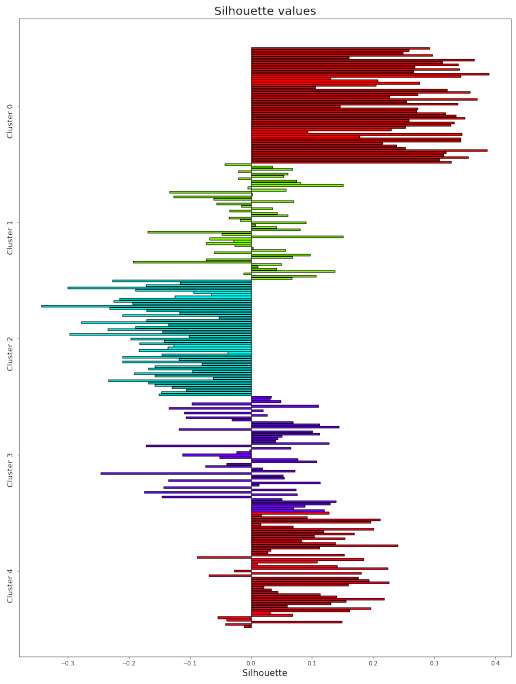

Figure 7. Silhouette score of a terrible partition (for reference; $DBI \approx 3.5$, $sil \approx -0.1$)

```
# parameter grid
partitions = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
new_dimensions = [2, 3, 4, 5, 6, 7, 8]
print("Grid search on {:d}x{:d} grid.".format(len(partitions), len(new_dimensions)))
```

```
Grid search on 11x7 grid.
```

```
# results dataframe
scores = pd.DataFrame(columns=['Dimension', 'Algorithm', 'Components', 'Davies-Bouldin-L1
', 'Davies-Bouldin-Mahalanobis', 'Silhouette-L1', 'Silhouette-Mahalanobis'])
scores
```

	Dimension	Algorithm	Components	Davies-Bouldin-L1	Davies-Bouldin-Mahalanobis	Silhouette-L1	Silhouette-Mahalanobis

```
data_raw = profiles_raw.to_numpy()
start_time = time.time()
for new_dim in new_dimensions:
    for nb_comps in partitions:

        print(">>>>><<<<<")
        print("Dimension is {:d}; looking for {:d} clusters.".format(new_dim, nb_comps))
        data = FA_full(data_raw, new_dimension=new_dim)
        profiles = pd.DataFrame(data)
        print("Factor analysis done.")
        kmeans_row = Kmeans_full(nb_comps, data, profiles)
        gm_row = GM_full(nb_comps, data, profiles)
        print("Clustering is done.")
        scores = scores.append(kmeans_row)
        scores = scores.append(gm_row)
        print("<<<<<>>>>>")
        print()

total_time = time.time() - start_time
```

```
>>>>>>>>>>
Dimension is 2; looking for 2 clusters.
Factor analysis done.
Clustering is done.
```

Figure 8. Grid search for FA

```
# parameter grid
partitions = [4, 5, 8, 10]
new_dimensions = [2, 3, 4, 5, 6, 7, 8]
print("Grid search on {:d}x{:d} grid.".format(len(partitions), len(new_dimensions)))
```

```
Grid search on 4x7 grid.
```

```
# results dataframe
scores = pd.DataFrame(columns=['Dimension', 'Algorithm', 'Components', 'Davies-Bouldin-L1
', 'Davies-Bouldin-Mahalanobis', 'Silhouette-L1', 'Silhouette-Mahalanobis'])
scores
```

	Dimension	Algorithm	Components	Davies-Bouldin-L1	Davies-Bouldin-Mahalanobis	Silhouette-L1	Silhouette-Mahalanobis

```
X = profiles_raw.to_numpy()
X = np.round(X, 3)
start_time = time.time()
for new_dim in new_dimensions:
    print(">>>>>>>>>>")
    # encoder training and dimensionality reduction
    autoencoder = Auto_Encoder((22, ), new_dim, activation='sigmoid',summary=False)
    autoencoder.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001), loss=tf.keras.losse
s.MSE)
    autoencoder.fit(X, X,
        epochs=100,
        batch_size=128,
        shuffle=True,
        validation_split=0.2)

    encoder = Model(autoencoder.input, autoencoder.get_layer('bottleneck').output)
    data = encoder.predict(X)
    profiles = pd.DataFrame(data)
    print("Autoencoder is trained.")

    for nb_comps in partitions:

        print("Dimension is {:d}; looking for {:d} clusters.".format(new_dim, nb_comps))
        # clustering
        kmeans_row = Kmeans_full(nb_comps, data, profiles)
        gm_row = GM_full(nb_comps, data, profiles)
        print("Clustering is done.")
        scores = scores.append(kmeans_row)
        scores = scores.append(gm_row)
        print("<<<<<>>>>>")
        print()

total_time = time.time() - start_time
print("Grid search is over.")
```

```
>>>>>>>>>>
Train on 4349 samples, validate on 1088 samples
Epoch 1/100
```

Figure 9. Grid search for Autoencoder

Annex B

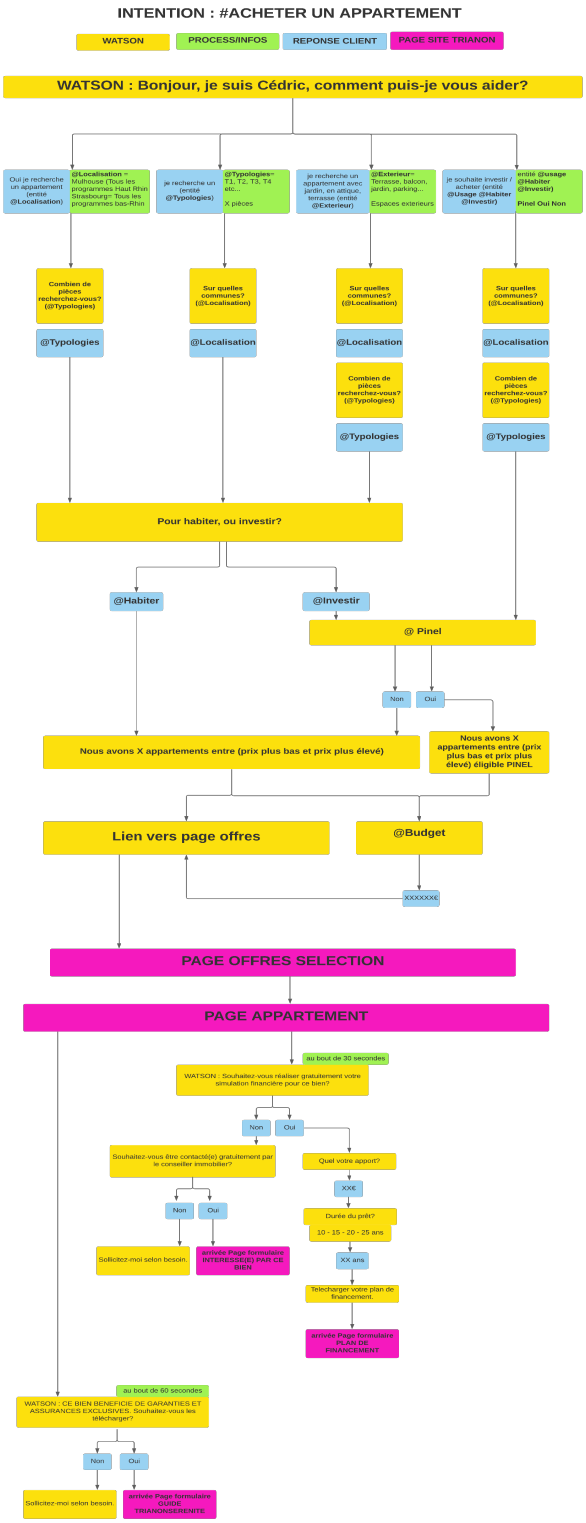

Figure 1. Draft of the next version of a chatbot

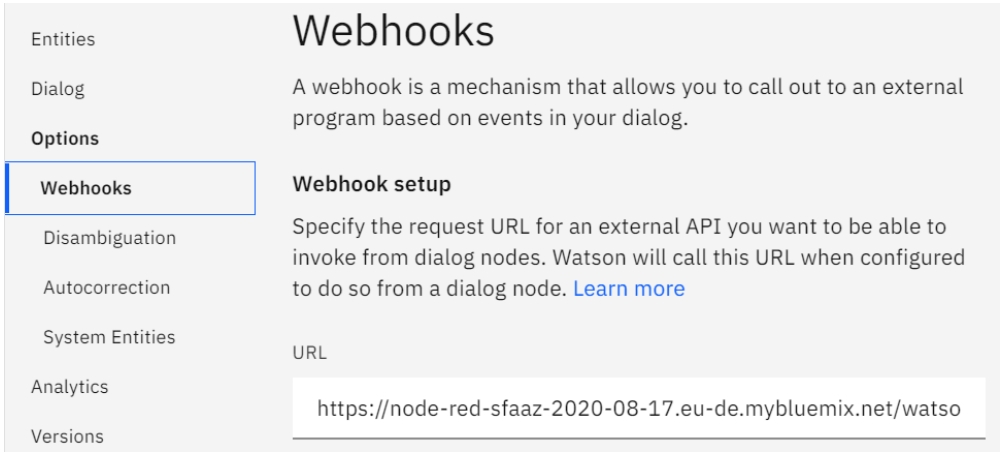

Figure 2. Webhooks setup in Watson Assistant

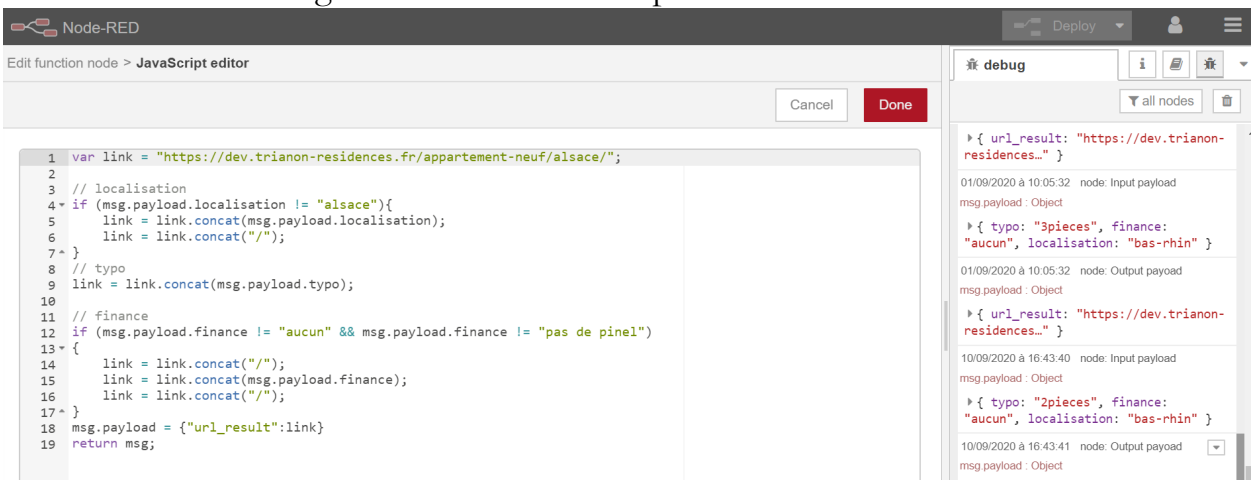

Figure 3. "Site_URL_Generator" javascript code

Annex C

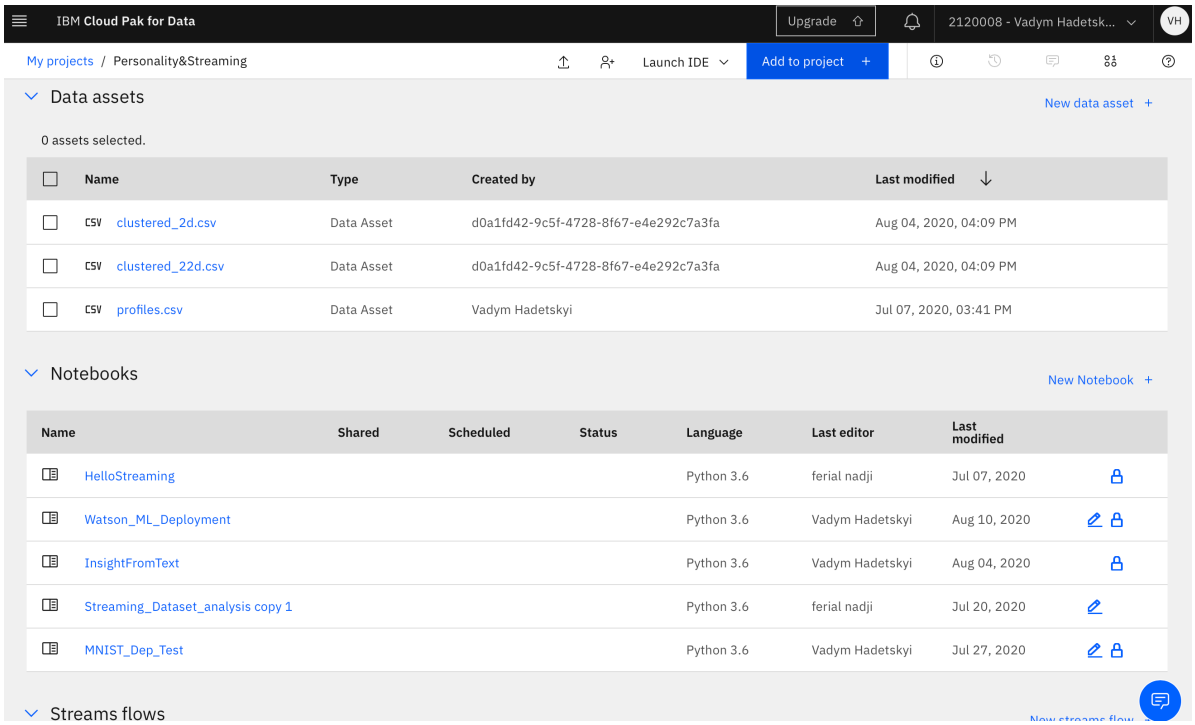

Figure 1. IBM Watson Studio

```python
# custom
import app_func.aux_func
from app_func.aux_func import get_insight_local

# flask
from flask import Flask, send_file
from flask import request, render_template, jsonify
IMG_FOLDER = "static/images/"

app = Flask(__name__)
# app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 0
app.config['UPLOAD_FOLDER'] = IMG_FOLDER
matplotlib.use('agg')

@app.after_request
def add_header(r):
    r.headers["Cache-Control"] = "no-cache"
    return r

@app.route('/')
def home():
    if os.path.exists(IMG_FOLDER+'radar_plot'):
        os.remove(IMG_FOLDER+'radar_plot')
    return render_template('home.html')
```

```python
@app.route('/joined', methods=['GET', 'POST'])
def my_form_post():
    text_input = request.form['text_input']
    word = request.args.get('text_input')
    prediction = get_insight_local(text_input)
    response = 'prediction_{:d}.html'.format(prediction)
    print(response)
    return render_template(response)


if __name__ == '__main__':

    app.run(host='0.0.0.0', debug=False)
```

Figure 2. Personality Identification executable (app.py)